فيجوال بيسك

من البداية حتى قواعد البيانات

VISUAL BASIC

FROM BEGINHNG TO DATABASES

أ. د. محمد الفيومي

أستاذ المحاسبة والمراجعة

كلية التجارة جامعة الإسكندرية

مقدمة

البرمجة المرئية هي أسلوب جديد للبرمجة ، تستخدم فيه برامج مساعدة لتصميم واجهة الاستخدام المكونة من الأزرار والنصوص والقوائم وربطها بالكود، وتسمى هذه البرامج المساعدة ببيئة التطوير المدمجة **Integrated Development Environment IDE**.

وكانت البرامج فيما مضى تستخدم ما يسمى بسطر الأوامر **Command Line** فكان يظهر البرنامج في صورة عدة أسطر ثم يتوقف ليطلب مثلا أدخال الأسم ثم الضغط على مفتاح **Enter** ثم ينتظر لتدخل الاسم ثم يكمل عمله، أما الآن ومع الواجهات الرسومية تظهر أمام المستخدم عشرات الأزرار والخيارات والقوائم وغيرها ولا يمكن للبرنامج أن يتوقع ما الذي سيحدث في الخطوة التالية، لذا فإن البرنامج يقسم إلى عدة وظائف ينفذ كل منها عندما يحدث ما يسمى بالحدث **event** نقرة الزرار مثلا تعتبر حدثا، ضغط أحد المفاتيح يعتبر حدثا، الاتصال بالإنترنت يعتبر حدثا، كل هذه تعتبر أحداث، وتسمى الدالة التي تعمل عند حدوث الحدث بالدالة الحدثية أو الدالة المرتبطة بالحدث .

ومن أشهر بيئات التطوير الرسومية

- **Visual C++**
- **Visual Basic**
- **Visual J++**
- **Delphi**
- **Borland C++**
- **Borland C++ Builder**
- **Java Builder**

وتستخدم هذه البرامج نسخ محسنة من لغات البرمجة العادية والقديمة وتدمجها في بيئة التطوير الخاصة بها لذلك فإن **Delphi** مثلا ليست لغة برمجة بمعنى الكلمة، وإنما هي بيئة تطوير تستخدم لغة محسنة من **Pascal** تتميز بميزات الكائنات وميزات أخرى فيطلق عليها لغة **Delphi** .

قد تبدو نافذة الفيجوال بيسك مليئة بعض الشيء ولكنها سهلة الاستخدام وسنتعرف على كل جزء من أجزاءها بالتفصيل.

هذا الكتاب موجه لطلاب شعبة نظم المعلومات وطلاب الدراسات العليا بكلية التجارة جامعة الإسكندرية ولكافة المهتمين بدراسة لغات الحاسب.

نسأل الله التوفيق.

أ. د. محمد الفيومي محمد

أستاذ المحاسبة والمراجعة

كلية التجارة جامعة الإسكندرية

الفصل الأول

مدخل لهياكل البيانات

يتضمن ميدان الحاسبات الاليكترونية ثلاثة ميادين رئيسية هي: البرامج Software، والآلات Hardware، والأفراد Personnel.

يختص الميدان الأول بدراسة كيفية التعامل مع الحاسب والبيانات وهو ميدان البرمجة، ويختص الميدان الثاني بدراسة الوحدات الآلية بالحاسب، أي وحدات الحاسب المختلفة وصيانتها، بينما يختص الميدان الثالث بالعاملين بإدارة الحاسب وتخصصاتهم ومؤهلاتهم.

والموضوع الرئيسي لهذا الكتاب هو البرمجة بلغة فيجوال بيسك Visual Basic، لذلك سيتم التمهيد لموضوع البرمجة Programming.

البرنامج Program:

هو مجموعة من الأوامر الموجهة للحاسب لتنفيذ عملية معينة، في خطوات محددة، وتسلسل منطقي يربط بين هذه الخطوات.

ويعد البرنامج لمعالجة مشكلة معينة، يستخدم فيها الحاسب كأداة رئيسية لتنفيذ الحل. ويمكن أن يرد الحل في صيغة عامة في شكل خطوط عريضة لخطوات الحل دون التعرض لتفاصيلها. أو أن يرد في شكل تفصيلي نتعرض فيه لكافة التفاصيل المطلوب إنجازها.

ويسمي الشكل الأول الأسلوب الخوارزمي Algorithmic Mode ويقوم علي مبدأ تجزئة الحل إلي خطوات إجمالية دون التقيد بالتفاصيل، ويجب توافر بعض الخصائص لهذا الأسلوب:

أ – تتابع الخطوات: فوفق هذا الأسلوب، يجزأ الحل إلي خطوات متعاقبة ذات ارتباط بينها، بحيث تؤدى إلي الغرض المطلوب.

ب – تحديد نقطتي بدء وانتهاء كل عملية من عمليات الحاسب: فلا تبدأ عملية بدون نقطة دخول **Enter Point** ولا تنتهي عملية بدون نقطة خروج **Ending Point**.

جـ– المرونة الكبيرة في الحل: فلا تحتوى الخطوات علي تفاصيل الخطوات، فالحل الموضوع يأخذ بعين الاعتبار المجاهيل والعموميات.

د – الربط المنطقي بين الخطوات.

و الشكل الثاني هو التفصيلي:أي خطوات البرنامج بالتفصيل، حيث نأخذ في الحسبان كافة التفاصيل، التي لم يتعرض لها الحل الخوارزمي. ويتناول البرنامج التفصيلي:

أ – الوحدات الآلية المستخدمة في البرنامج.

ب – الملفات المستخدمة في البرنامج لمعالجتها.

جـ– الدورات المستخدمة في البرنامج.

د – العمليات المختلفة التي كتب لأجلها البرنامج (مثل العمليات الحسابية الأربعة، عمليات المقارنة، المسح، التحويل، النقل...).

لغات البرمجة **Programming Languages**:

اللغة هي تلك الرموز والمصطلحات الخاصة للتعبير عن شيء ما، وذلك إذا ما استخدمت هذه الرموز والاصطلاحات وفقاً لقواعد معينة وأصول خاصة.

وبالنسبة للحاسب، فإن لغة الحاسب – أو لغات البرمجة – ترتبط بتعريف البرنامج نفسه.

فالبرنامج عبارة عن سلسلة من الأوامر والتعليمات الموجهة للحاسب، ولغة البرنامج هي: تلك الرموز والاصطلاحات الخاصة للتعبير عن الأوامر والتعليمات التي تستخدم وفقاً لقواعد وأصول معينة.

ولقد تطورت لغات البرمجة تطوراً كبيراً منذ ظهور أول لغة من لغات البرمجة، ومرت بعدة مراحل هي:

لغة الآلة Machine Languages:

أول لغة استخدمت في الحاسب كانت لغة الآلة وهي الشكل الذي تخـزن بـه البيانات داخل ذاكرة الحاسب، وتعتمد علي النظام الثنائي (٠، ١)، فكان يعـبر عـن الأوامر وكافة التعليمات بالرقمين الصفر والواحد، وكانت الأوامر والتعليمات تبـدو في صورة سلسلة طويلة من الأرقام الثنائية، وكان ينجم عن ذلك:

- ضياع وقت المبرمج في كتابة خطوات مطولة.
- زيادة احتمال وقوع الأخطاء نتيجة للاقتصار علي الرمزين (٠، ١) فمثلاً، لـو أراد مخطط البرامج إضافة محتوى المخزن رقم ٢ في الذاكرة إلي محتويات المجمـع يكون الأمر:

0110010000000010

العنوان رمز عملية الجمع

وأي خطأ من المبرمج أو في تسجيل البيانات كتبديل رقم صفر برقم ١ في موضع ما، كان يتسبب في تغيير المعني المستهدف من الأمر.

علي أن ميزة الكتابة بهذه اللغة هي انخفاض الوقت المستهلك في التنفيذ عن أي لغة برمجة أخرى. والعمليات بهذه اللغات تكتب كلها بأرقام ثنائية فمثلاً:

عملية الجمع	رمزها	٠٠١١١٠١١
عملية الضرب	رمزها	١١٠١١٠١١
عملية الطرح	رمزها	٠١١١٠٠٠١
عملية القسمة	رمزها	١١٠١١١١٠

ولتلافي الصعوبات في الكتابة بهذه اللغة تم تطويرها إلي لغة أخرى هي:

اللغات الرمزية Symbolic Languages:

أستعيض في هذه اللغة عن مخاطبة الحاسب مباشرة بالنظام الثنائي بلغة أخرى أكثر سهولة تستخدم الرموز العددية والرموز الأبجدية، وذلك للتعبير عن الأوامر الموجهة للحاسب، فمثلاً أصبحت رموز العمليات كما يلي:

أجمع	٢١
أطرح	٢٢
أقسم	٢٣
أضرب	٢٤

وتسمي هذه الرموز بالدليل الرقمي **Numeric Codes**.

ولصعوبة تذكر الأرقام أستعيض أيضاً عن لغة الأرقام بالأبجديات فمثلاً أصبحت رموز العمليات السابقة كما يلي.

أجمع ن	**Add N**
أطرح ن	**Sub N**
أقسم ن	**Div N**
أضرب ن	**Mul N**
أقرأ ن	**Read N**

وتسمي هذه الرموز بالرموز الأبجدية **Alphabetic Codes**، ويسمي النوعان الرموز الرقمية والأبجدية **Alphanumeric Codes**، وهي الخطوة الأولي لتبسيط التعامل مع الحاسبات.

ويجب أن يقوم الحاسب بترجمة هذه التعليمات الرمزية (الرقمية أو الأبجدية) إلي لغة الحاسب اى اللغة الثنائية.

والخطوة التالية في تسهيل عملية البرمجة، كانت في عدم ضرورة تحديد مكان تخزين كل قيمة بالضبط بل يكفي التأكد من أنها مخزنة في مكان ما بالذاكرة، وبذلك أصبح بالإمكان استخدام عنوان رمزي للتخزين **Symbolic Address** بدلاً من تحديد مواقع التخزين بدقة.

وتتم عملية ترجمة هذه التعليمات الرمزية بواسطة برنامج المترجم **Compiler** وهو برنامج مكتوب بلغة الآلة وظيفته تحويل التعليمات من الشكل الذي كتبت به إلي لغة تفهمها الآلة.

لذلك، فالبرنامج الأساسي ينتج عنه برنامج آخر مكتوب بلغة الآلة، يمكننا تشـــغيله وإدخال البيانات إليه.

أي أن كل برنامج يمر بمرحلتين:

١ – مرحلة ترجمة تعليمات البرنامج إلي لغة الآلة.

٢ – مرحلة تنفيذه.

ولقد وجد الباحثون، أن هناك طولاً غير مبرر في كتابة البرامج وأنه من الممكن تقليل احتمال وقوع الخطأ إذا ما استخدمت لغة إنجليزية قريبة من اللغة المتداولة كوسيلة لكتابة البرامج.

فظهرت المرحلة الثالثة من لغات البرمجة:

اللغات المتطورة Eveloted Languages:

وتنقسم اللغات المتطورة إلي نوعين:

١ – اللغات العامة Universal language:

وهي اللغات التي يمكن استخدامها في معظم الحاسبات، ومن أمثلتها:

COBOL – FORTRAN – PL / 1 – ALGOL – PASCAL ويسمي هذا النوع باللغات العليا High - Level Languages:

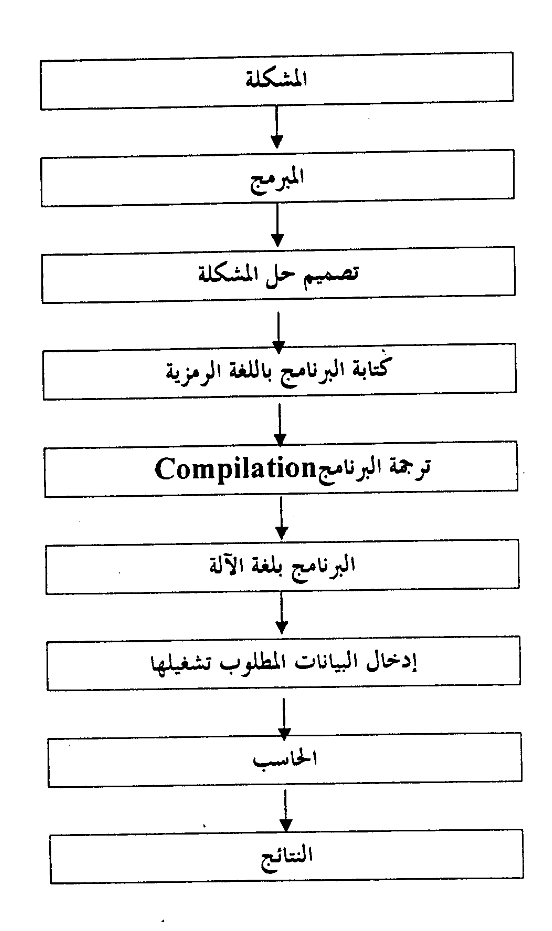
٢ – اللغات المحلية Local Languages:

وهي اللغات المستخدمة في حاسب معين دون غيره مثل:

PLAN علي حاســبات ICL، VERT، NEAT 3, Assembler علي حاسبات NCR.

ولكل لغة من هذه اللغات مزاياها وخواصها، فهناك لغات ذات مرونة كبيرة في بعض المجالات دون المجالات الأخرى.

ولضرورة نقل تعليمات البرنامج إلي لغة الآلة وجدت مترجمات لكل لغة من اللغات، مترجمات للغات العامة، ومترجمات للغات المحلية ويظهر الشكل التالي خطوات إعداد وتشغيل البرامج:



خطوات البرمجة للغات الرمزية

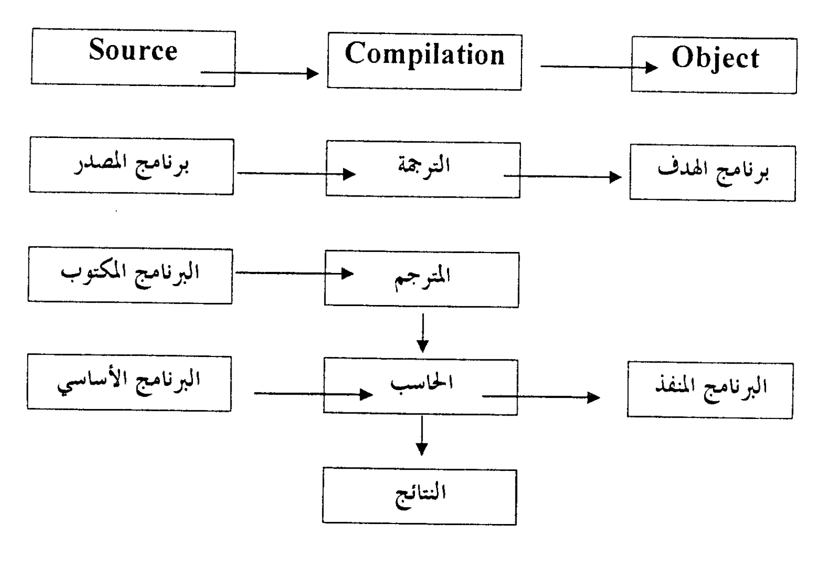
ترجمة البرنامج Compilation

بعد انتهاء مصمم البرنامج من كتابة برنامجه بأي لغة، يجب إيصال هذه التعليمات والأوامر إلي الحاسب وذلك بواسطة وحدة إدخال مثل لوحة المفاتيح أو من أحد أوسـطة التخزين مثل الأشرطة أو الأسطوانات الممغنطة أو الأسطوانات المضغوطة أي إدخال هذه الأوامر إلي الحاسب.

والمرحلة التالية هي ترجمة هذه الأوامر والتعليمات إلي لغة الآلة باستخدام برنامج الترجمة **Compiler**.

وبعد انتهاء الترجمة، ينشيء برنامج جديد هو ذاته البرنامج الأصلي الموجـه إلي الحاسب ولكن بعد ترجمته إلي لغة الآلة.

ويسمي البرنامج الجديد بالبرنامج المنفذ **Executor** أو الهدف **Object**. أما البرنامج الأصلي قبل ترجمته فهو البرنامج المصدر **Source**.



ويتم إنشاء البرنامج المنفذ **Object** عندما يكون البرنامج خالياً من الأخطاء تماماً أي **Free of Errors**، وهذه الأخطاء علي نوعين من التأثير فهناك:

١ – الأخطاء التنفيذية **Executing Errors**:

وهي الأخطاء التي لا يمكن للبرنامج أن يكون قابل للتنفيذ عند حدوثها.

٢ – الأخطاء التحذيرية **Warning Errors**:

وهي جملة الأخطاء التحذيرية – لفت النظر – التي لا يمكن للبرنامج تجاهلها.

وهذه الأخطاء علي نوعين أيضاً هما:

أ – الأخطاء الشكلية **Format Errors**:

وهي الأخطاء الناتجة عن الأخطاء الإملائية في كتابة التعليمات.

ويمكن أن تكون هناك أخطاء نتيجة خطأ في القواعد **Syntax Errors**، وذلك نتيجة لعدم إتباع قواعد كتابة البرنامج.

ب – الأخطاء المنطقية **Logical Errors**:

وهي الأخطاء الناتجة عن عدم إتباع قواعد المنطق في كتابة التعليمات ولا يكتشفها الحاسب لأنها ترتبط بالمنطق والخطوات اللازمة للوصول إلى النتائج التي يرغب المبرمج في الحصول عليها.

البيانات **Data**:

هي تمثيل رمزي للحقائق.

أنواع البيانات **Data Types**: تقع البيانات المطلوب معالجتها في ثلاثة أنواع:

١ – البيانات الأبجدية **Alphabetic Data**: وهي كافة الحروف الأبجدية الإنجليزية حصراً وهي:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

ويبلغ تعدادها ٢٦ حرفاً والحروف العربية ٢٨ حرفاً.

٢ – البيانات الرقمية **Numeric Data**: وهي كافة الحروف الرقمية وهي حصراً:

٠ ١ ٢ ٣ ٤ ٥ ٦ ٧ ٨ ٩

٣ – البيانات المختلطة **Alpha / Numeric Data**: وهي الرموز والحـــروف المختلطة وذلك إضافة إلي بعض الرموز الخاص **Special Characters** مثـــل تلك المستخدمة في العمليات الحسابية أو الوصل أو الربط مثل:

+ – /^* ( ) ، ! = > <،؟.

ويبلغ تعداد هذه الرموز في بعض الحاسبات ١٧ رمزاً وفي بعضـــها الآخـــر ٢٧ حرفاً.

• المعلومات **Information**:

المعلومات هي البيانات بعد تشغيلها أي جمعها أو فرزها أو تلخيصها.

• معالجة البيانات **Data Processing**:

تتميز البيانات المطلوب معالجتها بأكثر من نوع وأهمها:

١ – بيانات التعريف: وهي تلك الرموز الخاصة بتمييز السجلات أو رموز الاســـتخدام وغالباً ما تكون هذه الرموز رقمية (مميزات).

٢ – بيانات الثوابت: وهي تلك البيانات الخاصة بالمقادير مثل الكيلو واللتر والطن.

٣ – بيانات الدلائل: كالاسم، العنوان...... الخ.

٤ – بيانات رقمية: كقيمة الرصيد، وقيمة الرصيد الدائن، والرصيد المدين وغيرها وهي بيانات رقمية تمثل قيماً لهذه المبالغ.

• تشغيل البيانات:

• عندما يتعامل الحاسب مع البيانات، فإنه يتعامل مع الأرقـــام (أو محتويـــات الحقـــول) بصرف النظر عما يعنيه هذا الرقم بالنسبة للمستخدم. فمثلاً، الرقم ١٧٢ قـــد يكـــون طول شخص، أو مسافة بين مدينتين، أو سعراً لكمية من البضاعة، أو كمية لسلعة ما، أو رمزاً لأي شيء آخر.

وعلي هذا الأساس لو أدخلنا للحاسب البيانات التالية:

١٢٣٤٥٦٧٨٩٠

فقد تكون الخانة الأولي مشيرة لرمز الجنس وهنا ذكر. والخانتان اللتان بعـــدها تدلان علي العمر، والثمانية خانات الأخرى تشير إلي أربع درجات لأربع مواد، وكـــل

درجة بخانتين اثنتين. وهذه الرموز تمثل شيئاً هاماً بالنسبة لصاحب العلاقة وهو هنا المبرمج أو صاحب المشروع أو محلل النظام مثلاً.

وقد يبني المبرمج كافة علاقاته ونتائج أعماله علي مدلولات هذه الأرقام، لكن بالنسبة للحاسب فإنها ليست أكثر من مجموعة أرقام أو رموز رقمية يتفاعل معها أو مطلوب منه التفاعل معها، ثم إخراج نتائج هذه المعاملة إلي مستخدم الحاسب.

ويجب أن نتحرى الدقة حين إدخال البيانات للحاسب، علي المبرمج أن يكون مدركاً تماماً لما تعنيه هذه الرموز.

وفي كل الأحوال:

١ – فلكي تنفذ البرامج يجب إدخالها إلي ذاكرة الحاسب أولاً.

٢ – يتم تنفيذ التعليمات – تعليمات وأوامر البرنامج – داخل الحاسب بصرف النظر عما تعنيه هذه التعليمات والحاسب للمبرمج.

لن يقبل التعليمات والأوامر ما لم يتوفر فيها الشروط المطلوبة والنمط، من حيث القواعد والشكل والمنطق.

الملفات Files:

الملف **File** هو كافة البيانات المطلوب معالجتها، أو هو مجموعة من السجلات المرتبطة. فالبيانات الخاصة بفرد أو بواقعة معينة تسمي سجل. فمثلا لو كان المطلوب تجميع المعلومات التالية عن طلاب شعبة نظم المعلومات.

الاسم	العمر	الوزن	الطول	المهنة	المؤهل
عمر	٢٠	٦٨	١٦٨	–	ثانوية
علي	٢٢	٦٨	١٧٠	–	ثانوية
نصر	٢٧	٦٩	١٧٢	موظف	جامعية
عزة	٢٥	٧٠	١٧٠	موظف	جامعية

سجل **Record**

حقل

ملــــف

فالبيانات الخاصة بفرد واحد هي سجل **Record** ولما كان الملف هو مجموعة من السجلات فإن:

File = All of Record

و السجل ذاته يحتوى علي بيانات خاصة بمتغيرات معينة، كالاسم، العمر، الطول، المهنة، والثقافة ويسمي المتغير الواحد بالحقل **Field**.

فالسجل الواحد هو مجموعة من الحقول المرتبطة:

Record = Group of Fields

إذن:

File = All of Record
Record = Group of Fields
Field = One or More Character

أي أن الملف هو مجموعة من السجلات.

و السجل مجموعة من الحقول.

و الحقل حرف واحد أو أكثر من الحروف. وهذا الحرف قد يكون أحد أشكال البيانـــات التي سبق ذكرها.

و الحقل له شكلين فهناك:

- الحقل الفئوي: (من الفئة) **Group of Fields**
- الحقل الأولي: **Elementary Field**.

و الحقل الفئوي: هو الحقل القابل للانقسام إلي حقول أصغر منه فمثلاً

Date
Day **Month** **Year** ينقسم إلى

فحقل التاريخ **Date** حقل فئوي تحته ثلاثة حقول فرعية هي اليوم والشهر والسنة.

و الحقل الأولي: هو الحقل الذي لا يقبل الانقسام إلي اصغر مما هو عليـــه. ففـــي مثالنـــا السابق، الحقول **Year** **Month** **Day** هي أصغر الوحدات في حقل **Date** وهي حقول أوليه.

و لمعالجة السجلات يجب معرفة كل من:

١ – كيفية التعرف علي السجلات.

٢ – كيفية معالجتها.

٣ – تكوين السجلات مع بعضها البعض لتشكل ملف.

و علي هذا يجب معرفة:

١ – الأشكال المختلفة للسجلات.

٢ – طرق الوصول إلي البيانات المسجلة علي هذه السجلات.

٣ – طرق تشكيل الملفات.

أشكال السجلات Records Types:

هناك معلومات يجب معرفتها عن السجلات وهي نوعين:

- المعلومات المتعلقة بالحقول التي يشكل منها السجل.
- المعلومات المتعلقة برتبة الحقل بالنسبة للأصل.

١ – بالنسبة للنوع الأول فإن المعلومات اللازمة هي:

١– اسم كل حقل علي حدة، ويطلق على أسم الحقل أي من:

Data name or Identifier or Field–Name or Reference

وهي أربعة تسميات لمضمون واحد.

٢ – ويجب معرفة طول الحقل أي أقصي عدد للحروف المشكلة لهذا الحقل.

٣ – معرفة نوع الحقل وشكل محتويات بياناته وهل هي من النوع الأبجدي أو الرقمي أو المختلط.

أما النوع الثاني من المعلومات فهي المتعلقة بالتتالي الهرمي للحقل، وإذا كان فنوياً أو أولياً والحقل من نوعين:

١ – حقل أولي.

٢ – حقل فنوي.

ومثال للنوع الأول:

	NAME	30 A / N
	AGE	2 N
حقول أولية	SEX	1 N
	LENGTH	3 N
حقل فئة	DATA	
	DAY	2 N
حقول أولية	MONTH	2 N
	YEAR	2 N

من الأمثلة السابقة الحقل الأولي هو حقل لا يندرج تحته أي حقل آخر والحقل الفنوي حقل يمكن أن يندرج تحته حقول أصغر منه.

أشكال السجلات:

السجلات ذات أكثر من نوع وأهم الأنواع هي:

١ – السجلات ذات الطول الثابت **FIXED LENGTH RECORD**.

٢– السجلات ذات الطول المتغير **VARIABLE LENGTH RECORD**

٢ – السجلات ذات الطول غير المعرف **UNDEFINED LENGTH RECORD**.

و السجلات من النوع الأول ذات طول ثابت موحد لكافة سجلات الملف الواحد، ولا يتغير من سجل إلي سجل.

و النوع الثاني من السجلات ذات أطوال متغيرة من سجل إلي آخر، وفي هذه الحالة يجب معرفة:

– الحد الأقصى للطول.

– الحد الأدنى للطول.

والنوع الثالث من السجلات. تكون البيانات فيها غير معرفة وليست ذات طول محدد ولتلافي خطورة هذه العملية، ولتطلبها الدقة لمعرفة أطوال السجلات فغالباً ما يتم تحويل هذا النوع من السجلات إلي النوع الثاني أي الطول المتغير.

أوسطه الملفات FILE MEDIA:

١ – أوسطه ممغنطة MAGNETIC MEDIAS.

٢ – أوسطه غير ممغنطة NON - MAGNETIC MEDIAS.

و البيانات المسجلة علي أوسطة مغناطيسية قد تكون علي:

– أشرطة ممغنطة.

– أسطوانات ممغنطة.

– اسطوانات مدمجة.

وفي الأشرطة المغنطة، فإن السجلات تكون محزمة في مجموعات تسمي الحزمة الواحدة BLOCK ويوجد فراغ بين هذه الحزم يسمي بالفجوات ما بين الحزم INTER BLOCK GAPS.

و السجلات المسجلة علي الشريط المغنط محددة بسجلين متميزين:

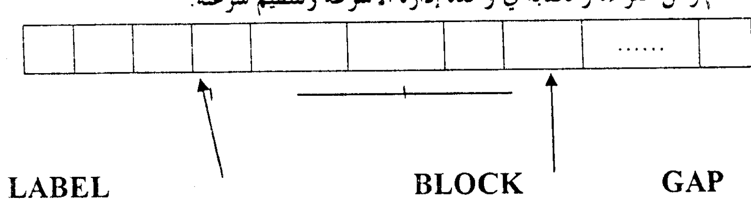
السجل الأول هو سجل بداية الشريط (الملف) ويسمي بـ LABEL ويحتوى علي ثلاثة معلومات هي:

١ – اسم الملف FILE NAME.

٢ – تاريخ إنشاؤه DATA.

٣ – رقم نسخته VERSION NUMBER.

و السجل الأخير هو سجل نهاية الشريط وهو TAPE END MARK ويوجد بين كل مجموعة سجلات فجوة الغرض منها تنظيم عملية تحرك الشريط وتوقفه أمام رأس القراءة والكتابة في وحدة إدارة الأشرطة ولتنظيم سرعته.



و لإيجاد عدد السجلات الموجودة في البلوك الواحد يتم تطبيق القاعدة التالية:

$$\text{BLOCK RECORD NUMBER} = \frac{\text{BLOCK LENGTH}}{\text{RECORD LENGTH}}$$

أي طول الحزمة مقسوماً علي طول السجل الواحد، يعطي عدد السـجلات في الحزمة الواحدة، أما طول الحزمة الواحدة فهو غير ثابت.

ففي بعض الحاسبات يبلغ هذا الطول ٢٠٤٨ حرفاً وفي أخرى يبلغ ٣٠٧٢ وفي ثالثة يبلغ ٤٠٩٦ حرفاً.

و بالنسبة للبيانات المسجلة علي اسطوانة ممغنطة والاسطوانات المدمجة فإن البلوك يماثله طول القطاع SECTOR وهذا الطول أيضاً غير موحد، ففي بعض الحاسـبات يبلغ الطول ٥١٢ حرفاً للقطاع الواحد، وفي بعضها الآخر يبلغ ٢٥٦ حرفاً، ولمعرفة عدد سجلات القطاع الواحد:

$$\text{NUMBER OF RECORDS IN A SECTOR} = \frac{\text{SECTOR LENGTH}}{\text{RECORD LENGTH}}$$

و بالنسبة للبواقي، وبواقي القسمة، سواء في الملفات أو في القطاع فإنها تضـاف آلياً إلي الفجوات بين البلوكات أو تترك فارغة حتى بداية قطاع جديد.

و بالنسبة للملفات علي الأوسطه غير الممغنطة، مثل الكشوف المطبوعـة، فـلا وجود لمفهوم البلوكات أو القطاعات وبالتالي فإن طول البلوك أو القطاع في هذه الحالـة مساوي تماماً لطول السجل.

أي:

$$\text{RECORD LENGTH} = \text{BLOCK LENGTH}$$

أي أن السجل هو ذاته لبلوك.

طرق تنظيم الملفات FILE ORGANIZATION :

هناك فرق بين تنظيم الملفات وطرق معالجاتها، ولا يجب الخلط بينهما، فتنظيم الملفات هو الشكل الذي سجلت فيه البيانات علي وسط التخزين، ويوجد أكثر من شكل:

التنظيم المتتالي SEQUANTIAL ORGANIZATION :

يعتبر التنظيم متتالياً إذا كانت السجلات مسجلة وراء بعضها البعض وفق ترتيب تسلسلي كترتيب هجائي أو رقمي مثلاً، بحيث لا يمكن الوصول إلي سجل ما دون المرور علي ما قبله من سجلات. والتتالي له شكلين فرعيين:

التتالي الطبيعي:

حيث يكون التتالي المنطقي مماثلاً لما عليه التتالي في مواقع التخزين، أي أن الترتيب المنطقي لهذه السجلات مماثلاً تماماً لترتيبها الطبيعي كما في الملفات المسجلة علي الأشرطة الممغنطة.

التتالي المنطقي:

ولا يتماثل فيه الترتيب المنطقي للسجلات مع الترتيب الطبيعي لها وطريقة الوصول إلي السجلات تتم وفقاً لتسلسلها المنطقي وليس لطرق وضعها علي الوسط. وهذا الأسلوب لا يكون إلا علي الأسطوانات الممغنطة والمدمجة.

التنظيم العشوائي أو الوصول المباشر RANDOM ORGNIZATION:

لا يتم هذا النوع إلا على الاسطوانات الممغنطة والاسطوانات المدمجة، ويلزم تحديد مفتاح للسجل وعنوانه على وسط التخزين وهذا الأسلوب في التنظيم له ثلاثة أشكال:

- التنظيم النسبي (التنظيم المتتالي) **RELATIVE**:

في هذا الأسلوب عنوان السجل معروف بترتيبه ومن هنا جاءت تسمية المتتالي داخل الملف وترتيبه الطبيعي مناظر لترتيبه المنطقي في هذه الحالة.

- التنظيم الفهرسى **INDEXED** :

في هذا الشكل من التنظيم يكون الملف مفهرساً على مستوى كل سجل أو بشكل عام على مستوى مجموعة سجلات. ويجب تكوين فهرس بحيث تكون لسجلاته ارتباطاً بين مميز السجل وعنوانه بالملف الاساسى.

ويسمح الفهرس بالوصول مباشرة إلى المجموعة التي يتبعها الملف الاساسي للوصـــول إلى السجلات متتالياً، في هذا الأسلوب يمكن أن يكون الملف:

- متتالياً
- مباشراً

- التنظـــيم بالتقســـيم SEGMINTATION PARTIONING DYNAMIC:

في هذا الأسلوب، هناك ملف اساسي واحد، قد يكون مفروزا أو لا، ويكون الملف مقسماً إلى أقسام كثيرة. وتكون الأقسام مفهرسة ومقسمة وتتفـــق هذا التقسيمات مع التقسيمات الطبيعية للتمييز بين المجموعات. وهنا يمكن الوصـــول مباشـــرة إلى القســـم المطلوب، وبعدها نستخدم الطريقة المتتالية داخل القسم.

الاوسطة المناسبة لتنظيم الملفات:

بالنسبة للتنظيم المتتالي فان الاوسطة المناسبة لهذا النوع هي:

- الأشرطة الممغنطة MAGNETIC TAPES
- الاسطوانات المغنطة والمدمجة MAGNETIC DISCS
- الكشوف المطبوعة PRINTER LISTING

وفيما عدا الاسطوانات الممغنطة والاسطوانات المدمجة، فان كافة الأوسطة لا تقبل إلا التنظيم المتتالي أما بالنسبة للتنظيم العشوائي أو المباشر فان الأوســـطة المقبولـــة هـــي الاسطوانات الممغنطة والاسطوانات المدمجة كوسط يقبل العنونة.

اشتمال السجل المنطقي الواحد على أكثر من سجل طبيعي:

الشكل الطبيعي أن يكون طول السجل منطقياً هو ذاته طوله الطبيعي على انه في بعض الأحيان يصدف أن يوضع السجل الواحد المنطقي على أكثر من سجل.

طرق الوصول إلى السجلات:

ترتبط طرق الوصول إلى السجلات بتنظيم الملفات ارتباطاً وثيقاً. وهذه الطرق هي:

- الطريقة المتتالية **SEQUANTIAL ACCESS MODE**
- الطريقة المباشرة **DIRECT ACCESS MODE**
- طريقة الوصول على مراحل **DYNAMIC ACCESS MODE**

الطريقة المتتالية في الوصول للسجلات:

وفيها لا يمكن الوصول إلى سجل ما قبل المرور على كافة السجلات التي تسبقه ومن هنا جاءت التسمية " المتتالية " وهذا يعنى أن الملف مفروزاً على حقل ما، وأهم الأوسطة المتتالية هي الأشرطة الممغنطة، وكذلك يمكن استخدام الاسطوانات الممغنطة والمدمجة.

الطريقة المباشرة في الوصول للسجلات:

يتم فيها الوصول إلى السجل بصورة مباشرة وبشكل مستقل تماماً عن باقي السجلات الأخرى، وهذا لا يستدعى أن يكون الملف مفروزاً ولكن يخصص عنوان ليكون:

- معرف للسجل.
- مفتاح للسجل.

الطريقة التقسيمية للوصول للسجلات:

وهذه الطريقة مزيج للطريقتين السابقتين فتكون أقسام هذا الملف مفهرسة، ويمكن الوصول إلى أي قسم مباشرة، وبعد الوصول إلى القسم المطلوب تتم معالجة السجلات فيه اعتباراً من السجل الأول وبالطريقة المتتالية.

أشكال الملفات **FILE TYPES**:

تنقسم الملفات إلى شكلين رئيسين هما:

- ملفات الإدخال **INPUT FILES**
- ملفات النتائج **OUTPUT FILES**

ويتفرع عن هذين الشكلين كثير من أشكال الملفات منها:

- الملفات الأساسية **Master File**:

وهى الملفات الجاهزة للتشغيل والتنفيذ عليها، مثل:

- ملف العملاء في البنوك
- ملفات الموازنات.
- ملفات المخزون.

- ملف العمليات **Transaction File**

وهى الملفات التي تحدث الملفات الأساسية لنحصل في نهاية العملية على ملفات أساسية جديدة.

- الملفات التاريخية (التوثيقيـــة).
- ملفات الجداول.

عمليات معالجة الملفات:

يتم تنفيذ عديد من العمليات على الملفات للاستفادة منها وهى:

١- تحديث الملف الرئيسي:

هناك بيانات معينة في الملفات الرئيسية يتم تعديلها باستمرار لكي تحتوى على الدوام البيانات في أحدث صورها، ففي ملف المخزن يعدل رصيد المخزون بعد كل حركة صرف أو إضافة، كما يعدل ملف العملاء ليعكس آخر موقف له من حيث المديونيـــة. وتعـــرف عملية التعديل هذه بالتحديث **Updating** والتي يستعان في إجرائها بملفات المعاملات.

٢- تعديل الملف الرئيسي:

يتطلب الأمر إضافة سجلات جديدة باستمرار (كإضافة عميل جديد) أو محو سجل ما (حذف صنف مخزني توقفت الشركة عن استخدامه). ويطلق على عملية التعديل عادة صيانة الملفات **File Maintenance** وهى تختلف عن عملية التحـــديث، والـــتي تتناول البيانات الدائمة التغير بطبيعتها (المعاملات).

٣- الاسترجاع من الملفات:

يقصد بعملية الاسترجاع Retrieving عرض سجلاته من خلال حقل مفتاحي (عرض سجل الموظف رقم ١٢٣٤) أو بناء على قيمة معينة (عرض سجلات الموظفين المولودين قبل ١٩٨٥/١/١).

٤- ترتيب الملفات:

يمكن ترتيب Sorting الملفات منطقياً وفقاً لقيم بعض الحقول، فملف الموظفين مثلاً يتم تنظيم سجلاته بالنسبة للحقل المفتاحى الخاص برقم الموظف.. ولا تخضع ملفات المعاملات في العادة للترتيب، ولكن قبل تحديث الملف الرئيسي يتم ترتيب ملف المعاملات المتصلة به بنفس طريقة ترتيبه، فيرتب ملف ساعات العمل طبقاً لنفس ترتيب ملف الموظفين لكي يتم تحديث الملف الأخير بالنسبة للأجر.

والترتيب من العمليات الشائعة والهامة، لدرجة وجود برامج تطبيقية لإجراء التصنيفات القياسية على الملفات.

٥- دمج الملفات:

يمكن ربط ملفين أو أكثر مرتبين بنفس التسلسل في ملف واحد، ويسمى ذلك بالدمج Merging وقد يكون الدمج بحيث يضم الملف الجديد كل سجلات الملفات الداخلة في عملية الدمج، ويبين الشكل التالي مثالاً لذلك، حيث تم إنتاج ملف معاملات خاص بعدد ساعات العمل لإدارتين، وذلك بدمج ملفين يخص كل منهما إدارة منهما.

Branch A

Hours worked on day

emp#	1	2	3	4	5	6	7
1103	8	8	8	8	8	0	0
1125	8	8	0	0	0	0	0
1128	7	7	7	7	7	0	0

Branch B

Hours worked on day

emp#	1	2	3	4	5	6	7
1126	8	8	8	8	7	0	0
1127	8	8	8	8	8	6	6
1130	8.5	8.5	8.5	9	9	0	0

Branch A

Hours worked on day

emp#	1	2	3	4	5	6	7
1103	8	8	0	0	0	0	0
1125	8	8	0	0	0	0	0
1126	8	8	8	8	7	0	0
1127	8	8	8	8	8	6	6
1128	7	7	7	7	7	0	0
1130	8.5	8.5	8.5	9	9	0	0

دمج ملفين

كما يمكن أن يكون الملف المنتج بحيث يضم سجلات مكونة من أجزاء من الملفات المدمجة، ففي الشكل التالي أنتج ملف يضم كميات وأسعار الأصناف المخزنة من ملفين، الأول يضم سعر الوحدة. والثاني يضم الكميات المخزنة.

Stock Price	
٤٨٦٣	١,٠٣
٤٨٦٩	٦,٤٥
٤٨٧٠	٣٢,٥٦
٤٨٧٧	٣٢,٠٢

Stock #	Quantity on hand	Warehouse #
٤٨٦٣	٣٥	A
٤٨٦٩	٤٣	A
٤٨٧٠	٣	B
٤٨٧٧	٢١	C

Stock#	Quantity on hand	Warehouse #
٤٨٦٣	35	1.03
٤٨٦٩	43	4.45
٤٨٧٠	3	32.56
٤٨٧٧	21	23.02

دمج سجلين

مقارنة بين أوسطة تخـــزين الملفـــات: FILE SPECIFICATION COMPRISE

الأشرطة الممغنطة:

المزايا:

١- طاقتها كبيرة إذا قيست بالاسطوانات الممغنطة.

٢- وسط هام يستخدم كوسيط في تحويل البيانات ولحفظ نســـخ مـــن محتويـــات الاسطوانات الطلبة.

٣- اقل تكلفة.

٤- إمكانية إعادة الاستخدام لأكثر من مرة.

العيــوب:

١- تستخدم في الوصول المتتالي فقط.

٢- غير مرئية بالعين المجردة.

٣- سريعة التأثر بالمجالات المغناطيسية.

الأسطوانات الممغنطة والمدمجة:

المزايا:

١- طاقتها ضخمة في استيعاب الكثير من المعلومات.

٢- يمكن استخدامها كوسيلة للوصول المتتالي أو العشوائي.

٣- إمكانية تسجيل أكثر من ملف عليها.

العيــوب:

١- سهولة التعرض للخدش (خدش المسارات).

٢- غير مرئية مباشرة.

أنواع البرامج PROGRAM TYPES:

البرامج نوعان:

- برامج أساسية.
- برامج تطبيقية.

والبرامج ثلاثة أنواع من حيث الوظيفة:

- برامج إدخال **INPUT PROGRAMS**
- برامج منافع **OPERATING, UTILITY PROGRAMS**
- برامج نتائج **OUTPUT PROGRAMS**

١- برامج الإدخال:

يقصد بها البرامج الأولية الخاصة بتجهيز الملف وذلك استعداداً لتهيئته لإيصاله إلى مرحلة جاهزة للتنفيـــذ عليه.

واهم أنواع هذه البرامج:

برامج القراءة الأولية والطباعة الأولية EDITING

برامج التدقيق أو التصحيح VALIDATION

٢- برامج التشغيل (البرامج المساعدة) OPERATING

وظيفة هذا النوع من البرامج تسهيل عمليات تنفيذ أو إتمام المرحلة الثالثة والربط بين المرحلتين (مرحلة الإدخال ومرحلة استخراج النتائج) وهذا النوع يحتوى على:

- برامج الفرز SORTING
- برامج التحويل من وسط إلى وسط TRANSFARING
- برامج الدمج MERGING
- برامج النسخ COPYING

وهذا النوع من البرامج وان أمكن كتابته برمجياً إلا انه في معظم الأحيان تتولى معظم الشركات تجهيز هذه البرامج وإنزالها في مكتبة البرامج وما على الجهة الطالبة إلا استدعاء البرنامج المطلوب منها، دون الحاجة لإعادة برمجتها.

٣- برامج النتائج:

هي البرامج التي تقوم بمعالجة الملفات الخالية من الأخطاء لاستخراج النتائج النهائية من جداول وتطبيقات مختلفة أهمها:

- برامج طباعة الجداول LISTING
- استخراج ملفات جديدة من ملفات موجودة EXTRACTING
- تحديث الملف UPDATING
- إنشاء ملفات جديدة CREATING

وهناك تقسيماً آخر للبرامج من حيث الشكل:

- البرامج الرئيسية.
- البرامج الفرعية.

تدخل تحت هذا العنوان كافة البرامج التي ذكرناها باستثناء البرامج الوسيطة وحتى هذه الأخيرة يمكن أن تدخل تحت هذا البند إذا ما قام المبرمجين بكتابها.

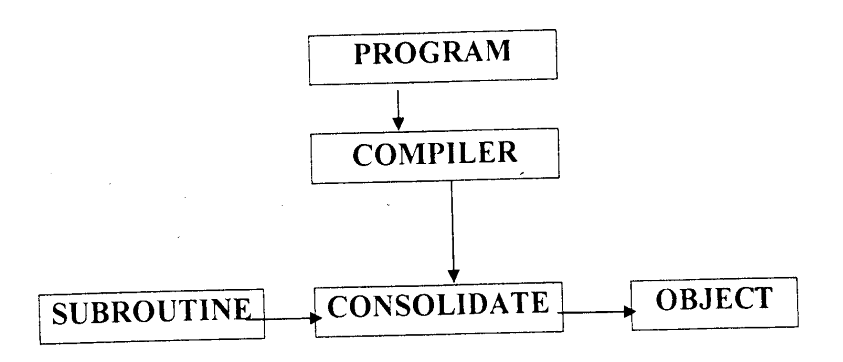
البرامج الفرعية SUBROUTINES :

تتكرر بعض التعليمات أكثر من مرة في كثير من البرامج خاصة في البرامج العلمية، لكن الاختلاف يقع فقط في المتغيرات أو المعلومات التي يتم عليها التشغيل، وكان يستدعى لاستخدامها إعادة طلبها في كل مرة وحجز أماكن لها في ذاكرة الحاسب ولكن لما كانت عملية الحجز هذه مكلفة اقتصاديا، لذلك تكتب الطريقة مرة واحدة، ويرجع إليها في كل مرة تحتاج إليها وتسمى هذه الأجزاء بالبرامج الفرعية SUBROUTINES

التجميع:

المقصود بهذه العملية، تجميع أجزاء البرامج الناتجة عن عملية الترجمة وإضافة أجزاء أخرى نحتاج إليها من مكتبة البرامج الفرعية الموجودة على الاسطوانة الممغنطة لكي يأخذ البرنامج صورته النهائية.

وتبدأ مرحلة التجميع بعد انتهاء المترجم من ترجمة الجزء المكتوب باللغة التي سيقوم بترجمتها، أما الأجزاء المكتوب بلغة الآلة فلا يحتاج إلى ترجمتها، بل تدخل مرحلة التجميع مباشرة.



برامج التشغيل OPERATING SYSTEM:

يقوم هذا النوع من البرامج بتشغيل الحاسب اعتماداً على مجموعـــة مـــن الأوامـــر والتعليمات الموجهة للحاسب تعطى عن طريق وحدات الإدخال مثل لوحة المفاتيح ومن ثم يتم تنفيذ هذه الأوامر والتعليمات.

وتسمى هذه البرامج OPERATING SYSTEM وتحتوى علـــى مجموعـــة البرامج التي تتولى عملية إدارة وتشغيل الحاسب.

وتحتوى برامج نظم التشغيل على:

١- برامج بدء التشغيل DAILY STARTING PROGRAMS:

تقوم هذه البرامج بإدخال البرنامج التنفيذي إلى ذاكرة الحاسب حيث يمكن بدء العمل بالتشغيل ويتم استدعاء هذا البرنـــامج بالضـــغط علـــى مفـــاتيح في لوحـــة التشـــغيل.

٢- البرنامج التنفيذي EXECUTE :

يقوم هذا البرنامج بما يلي:

- التحكم بجميع وحدات الحاسب.
- مراقبة تشغيله.
- التصرف أثناء حدوث أخطاء التشغيل.
- مراقبة تنفيذ برامج حل المشاكل.
- مخاطبة مراقب التشغيل كلما استدعت الضرورة ذلك.

٣- برامج المراقبة MONITOR:

تقوم هذه البرامج بمراقبة تشغيل برامج مستخدمي الحاسب وذلك للتخاطب بينه وبين البرنامج التنفيذي وكذلك التحكم في سير وتنفيذ العمليات طبقاً لتفصيلاتها المقدمة.

٤- برامج المنافع UTIL ROUTINES:

وهى مجموعة البرامج التي تقوم بخدمات عامة للتشغيل مثل تعديل مكتبات البرامج أو النسخ وغيرها.

تعددية البرامج MULTIPROGRAMING:

تولى جميع الحاسبات الحديثة تعددية البرامج أهمية خاصة، تؤدى تعددية تشغيل البرامج إلى استغلال وقت الحاسب استغلالا امثل، خاصة وان السرعة الكبيرة للحاسبات تعنى وجود وقت ضائع للحاسب، لان عمليات تغذية الحاسب بالبيانات لا تواكب سرعته.

ولتلاشى الوقت الضائع، يقوم الحاسب بتنفيذ أكثر من برنامج واحد بحيث إذا توقف تشغيل أحد البرامج، فإن برنامج المراقبة يحول وحدة التشغيل المركزية إلى البرنامج التالي في الترتيب لتقوم بتنفيذ تعليماته.

وهناك أولويات لتنفيذ البرامج فإذا توقف تنفيذ احد البرامج أنتظارا لقراءة سجل جديد مثلاً أو طباعة سطر، فان برنامج المراقبة يحول وحدة التشغيل المركزية إلى البرنامج التالي في الترتيب لتقوم وحدة التجهيز المركزية بتنفيذ تعليماته.

الفصل الثانى

مدخل للغة فيجوال بيسك

تتطور لغات الحاسب تبعاً لتطور أجهزة الحاسب وتقنياته وتطور أساليب البرمجة، ومن هذه اللغات لغة البيسك والتي ما زالت تمر بمراحل تطوير لتمكين المبرمجين من استخدام لغة بسيطة التركيب قوية العمل، ومن آخر تطورات هذه اللغة نجد لغة فيجوال بيسك **Visual Basic**.

ولغة فيجوال بيسك من اللغات التي طورت للعمل ببيئة ويندوز، وقد صممت ليستخدمها الأفراد العاديين الذين لهم إلمام بسيط ببرمجة الحاسب، كما أنها مصممة للمحترفين. ولقد غير نظام ويندوز المفاهيم التي كانت معروفة مسبقاً لمستخدمي الحاسب، وأدى إلى تغيير القواعد المعروفة للتعامل مع الحاسب. فالمبرمج الذي كان يستخدم لغات البرمجة التي تعمل بنظام دوس واجه مفاهيم جديدة عليه أن يتعلمها من البداية حيث لا يمكن تجاهل لغات ويندوز لأن هذا النظام أصبح مسيطراً علي معظم مستخدمي الحاسب.

وتتطلب هذه اللغة إلمام المستخدم بنظام ويندوز بالإضافة إلى إحدى لغات الحاسب مثل لغة باسكال أو كويك بيسك، لان معظم أوامر هذه اللغة مشتقة من لغة كويك بيسك. وللغة باسكال دوراً في الإعداد لاستخدام لغة فيجوال بيسك من ناحية إلمام المستخدم بالإجراءات **Procedure** والوظائف **Functions** حيث تعتمد معظم عمليات البرمجة بلغة فيجوال بيسك علي هذا الأسلوب في تصميم البرامج.

مفهوم البرمجة بلغة فيجوال بيسك

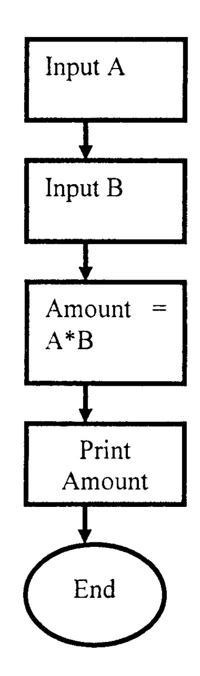
يختلف تصميم البرنامج بلغة فيجوال بيسك عن الطرق السابقة لتصميم البرامج. فمن المفاهيم الجديدة لبرنامج ويندوز تعدد المهام. أي إمكان تشغيل عدة برامج في نفس الوقت.

حيث يستطيع الحاسب خدمة عدة برامج في نفس الوقت، اى انه يمكن للحاسب معالجة الأوامر الواردة بعدة برامج في نفس الوقت حسب متطلبات العمل وليس حسب متطلبات البرنامج. ولنفهم هذه الطريقة، نفترض أن لدينا برنامجين يعملان في نفس الوقت، أحد البرامج به أمر لإدخال البيانات إلى الذاكرة، والبرنامج الآخر به أمر لعملية حسابية. سنجد أن نظام ويندوز ينفذ العملية الحسابية أثناء انتظار إدخال البيانات للبرنامج الآخر لكي تنجز لاحقاً، ذلك لأن المستخدم قد لا يرغب في إدخال البيانات في اللحظة المطلوبة، وبدلاً من الانتظار يستطيع نظام ويندوز البحث عن أمر آخر ليقوم بتنفيذه، بينما مسبقا إذا كان لدينا أمر لإدخال بيانات فان الحاسب لن يقوم بإنجاز أي أمر تالي إلى أن يتم إدخال البيانات بالأمر المحدد.

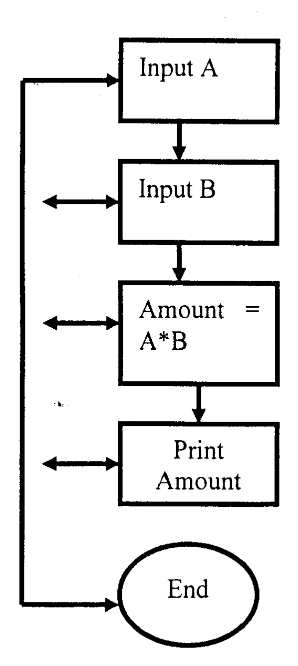
ويطلق علي تنفيذ الأوامر في البرامج التي تعمل تحت دوس اسم من اعلي إلى أسفل **Top Down** أي أن البرنامج سينفذ بدءاً من الخطوة الأولي إلى أن نصل إلى الخطوة الأخيرة دون القفز فوق الخطوات الآخري. أما تنفيذ الأوامر ببرنامج ويندوز فتعتمد علي الحدث **Event**، فالبرنامج لا ينفذ إلا بإصدار إشارة للأمر لكي يبدأ العمل، وهذه الإشارة ترسل إلى البرنامج بواسطة الفأرة أو لوحة المفاتيح.

ما يلي خريطة سير العمليات وأوامر لبرنامج ببيئة دوس وببيئة ويندوز، ويقوم البرنامج باستقبال أرقام عشوائية ثم ضربها وطبعها. فيبدأ باستقبال الرقم الأول ثم الرقم الثاني، ثم ضرب هذه الأرقام وطبع النتيجة ثم التوقف، أما في بيئة دوس فإن استقبال الرقم الثاني لن يتم ما لم يتم تنفيذ أمر استقبال الرقم الأول. والتوقف لن يتم إلا بعد تنفيذ كل الخطوات السابقة.

Programming
Procedures



البرمجة من اعلي لأسفل
Top-Down
Programming
(Dos)



البرمجة بالأحداث
Events
Programming
(Windows)

وما يلي أوامر البرنامج بلغة فيجوال بيسك

```
'Visual Basic Program Code

Dim a As Integer
Dim b As Integer
Dim c As Integer

Sub Command1_Click ( )
C= a * b
Iabel1.Caption = Str$(C)
End Sub

Sub Command2_Click ( )
End
End Sub

Sub Form _Load ( )
text1.Text =  " "
text2.Text =  " "
Label1.Caption = " "
End Sub

Sub Text1_Change ( )
A = Val (Text1.Text)
End Sub

Sub Text2_Change ( )
b = Val (Text2.Text)
End Sub
```

ويطلق علي أسلوب البرمجة ببيئة ويندوز مصطلح برمجة الحدث Events Programming أي أن الأوامر تنفذ بمعزل عن الأوامر الاخري ولا ترتبط معها

بالتسلسل الهرمي كما يحدث بالنسبة لأوامر البرمجة في بيئة دوس، والارتباط الوحيد بأوامر البرنامج هو في استقبال ومعالجة البيانات، أي أننا نستطيع استقبال الرقم الثاني ثم الـــرقم الأول ثم طباعة البيانات، كما نستطيع إنهاء البرنامج دون تنفيذ خطوات استقبال البيانات أو طبعها، اى أن كل أمر منفصل عن الأمر التالي ولن ينفذ إلا تحت تأثير حـــدث معـــين Event، والارتباط الوحيد بين هذه الأوامر هو استقبال ومعالجة البيانات. ويمكن تشبيه البرنامج بأماكن بها مجموعة من الأوامر، وكل أمر من هذه الأوامر ينفذ بناء علي تأثير من المستخدم وليس من البرنامج. ويتضح من خريطة سير العمليات الطريقة التي يعمـــل بهـــا البرنامج ببيئة ويندوز، حيث بامكانك التنقل بين الأوامر حسب الرغبة في تنفيـــذ الأمـــر المطلوب، فمثلاً يمكن إدخال الرقم الأول ثم الثاني، وبعد ذلك إعادة الـــرقم الأول مـــرة أخري إذا تبين أن هناك خطأ في الإدخال، وهذه العملية غير متوفرة ببيئة دوس.

الخطوات الأساسية لتخطيط وتصميم البرنامج في فيجوال بيسك.

تتطلب البرمجة باستخدام لغة فيجوال بيسك بعض الخطوات الأساسية لتصميم البرنـــامج المطلوب، والتي يجب أن تطبق في كل مرة فيها بتصميم برنامج جديد وهي:

١- تصميم واجهة البرنامج User Interface

Form1

Alex Car Co.

Car price 000.00

calculation method

Interest rate 0.000

Monthly Payment

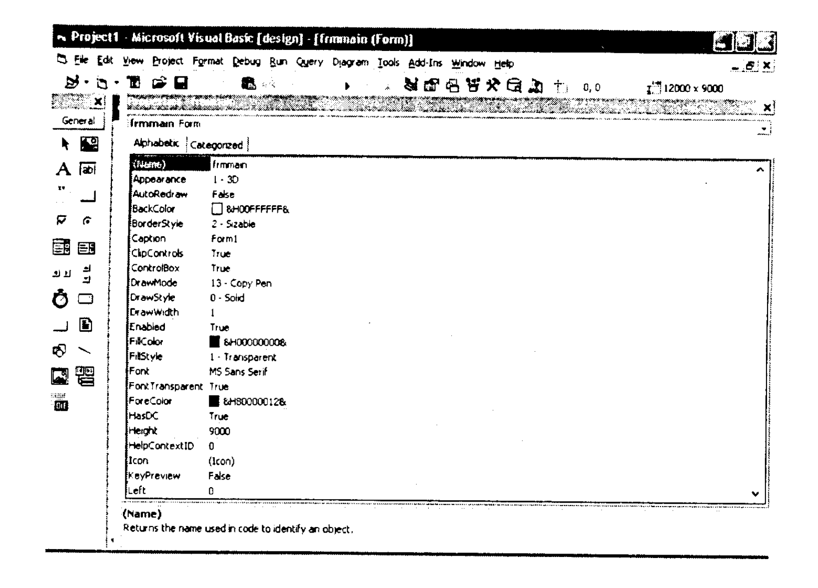
Total amount paid

No. of years 00

Calculate Exit

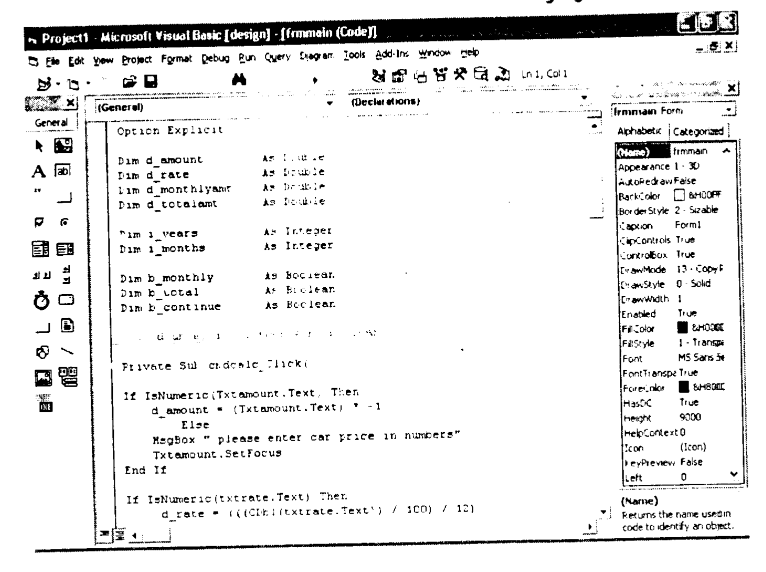
هذه الواجهة هي التي ستستعمل بين المستخدم والحاسب، وتشمل هذه الواجهة على شكل الشاشة – صناديق البيانات – القوائم – ازرار الأوامر أو الاختيارات – الصور – قوائم البيانات. ويمكن تصميم شاشة حوار بين المستخدم وبين الحاسب أو تصميم مجموعة شاشات بناء علي احتياجاتك.

٢- تحديد خصائص Properties الشاشة والأشكال التي وضعت داخل إطـــار الشاشة.



وتشمل الخصائص علي حجم الشاشة أو الشكل، اسم الشاشة أو الشكل، نـــوع اللـــون والخط، تحديد مواصفات خاصة تعتمد على نوع الشكل. ويتم في هذه المرحلة اســـتخدام نافذة الخصائص **Properties**.

٣- كتابة الأوامر:



يتم في هذه المرحلة كتابة الأمر المطلوب عند استخدام شكل معين، وكل شكل يمكن أن نكتب له مجموعة من الأوامر بناء علي أحداث متوقعة مسبقاً، وكل حدث يكتب بإجراء منفصل. بفرض أن هناك زراراً معيناً بالشاشة يمكن الانتقال إليه بواسطة الفأرة فإذا قمت بضغط الفأرة مرة واحدة فإن هذا يعنى أن هناك حدثاً مرتبط بعملية الضغط علي الفأرة مرة واحدة علي الزرار، أما إذا تم الضغط علي الزرار مرتين بالفأرة فهذا يعنى أن هناك حدثاً آخر مرتبطاً بهذه العملية، فالزرار يمكن الضغط عليه بالفأرة مرة واحدة أو مرتين وهذا يعنى بالنسبة للغة فيجوال بيسك أن لكل نوع من الضغط علي الزرار تأثيراً معيناً مختلفاً يتحدد بناء عليه العملية التالية بكتابة الأوامر المرتبطة بواسطة المبرمج، وبالتالي علي اللغة تنفيذ الإجراء المطلوب بناء على نوع الضغط علي الفأرة، وتعمل معظم برامج ويندوز باستخدام الفأرة في تنفيذ الأوامر. ويتم في هذه المرحلة استخدام شاشة المشروع **Project** لعرض وكتابة الأوامر.

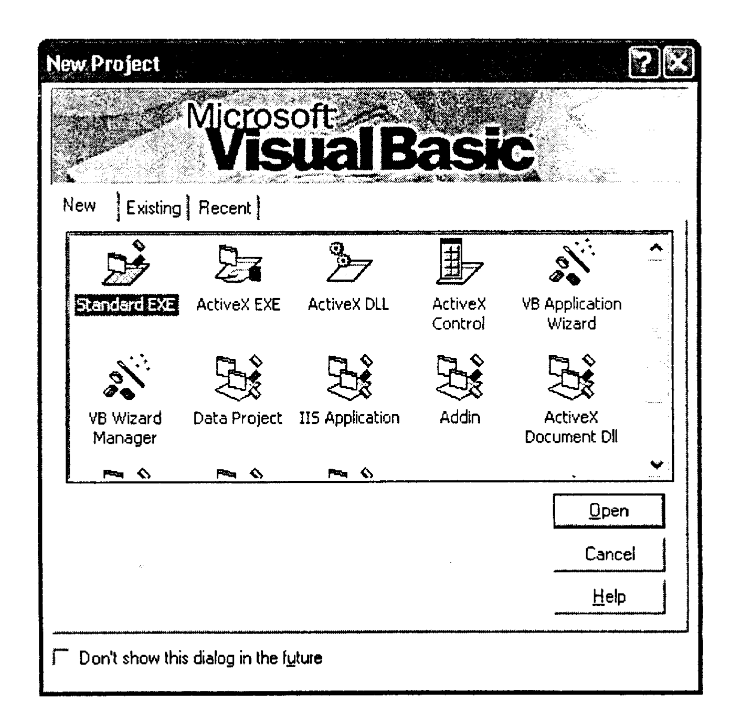
الفصل الثالث

بيئة البرمجة في فيجوال بيسك

تتطلب البرمجة بلغة فيجوال بيسك التعامل مع بيئة شاشات اللغة أولا ثم القيام بكتابة الأوامر ثانياً، ولهذا فإن الإلمام ببيئة اللغة سيساعد علي كتابة البرامج، لذلك سندرس أهم الشاشات المرتبطة بتصميم البرامج.

تشغيل البرنامج:

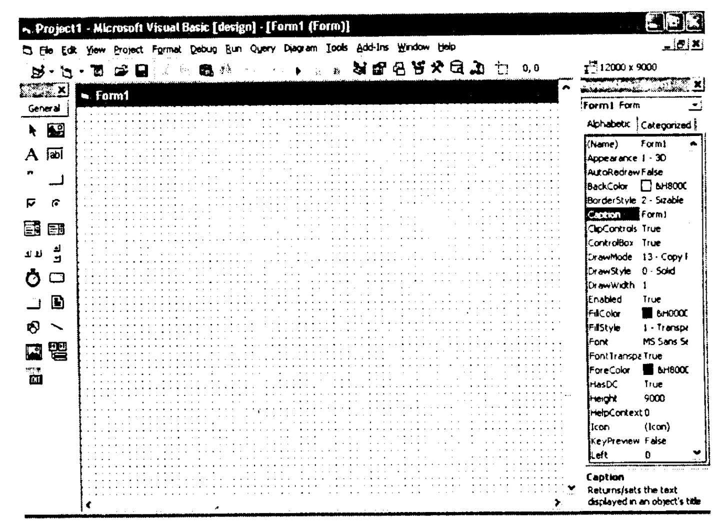
بعد تركيب لغة فيجوال بيسك سيكون البرنامج مجموعة خاصة به اسمها VISUAL BASIC.



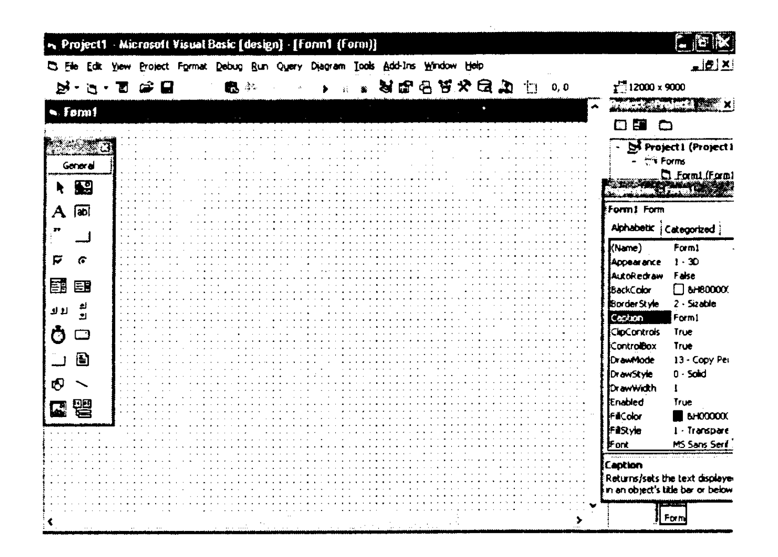
بهذه المجموعة عدد من الأيقونات، ولن نحتاج في البداية إلا إلى الأيقونة الـــتي تســـتخدم لتشغيل البرنامج. وعند تشغيل برنامج فيجوال بيسك يتم عرض خمس نوافذ علي شاشة الحاسب تظهر مع نوافذ البرامج المستخدمة سابقاً، لهذا عليك قبل تشغيل البرنامج اختيار قائمة الاختيارات من وحدة برنامج التحكم ثم حدد أمر تصغير أثناء التشغيل، لغلق كل النوافذ السابقة لتشغيل البرنامج أثناء عمل البرنامج مما يتيح لك معرفة النوافذ الخاصـــة ببرنامج لغة فيجوال بيسك.

لبدء تشغيل البرنامج والتعرف علي النوافذ التي ستستخدم أثناء تصميم البرامج نفذ مـــا يلي:

١- ضع الفأرة علي أيقونة برنامج فيجوال بيسك واضغط زرار الفـــارة مـــرتين متتاليتين.

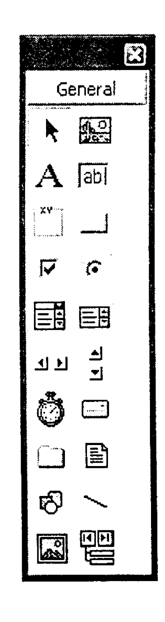


٢- ستظهر لك شاشة جديدة بها مجموعة من النوافذ، كما يلي:



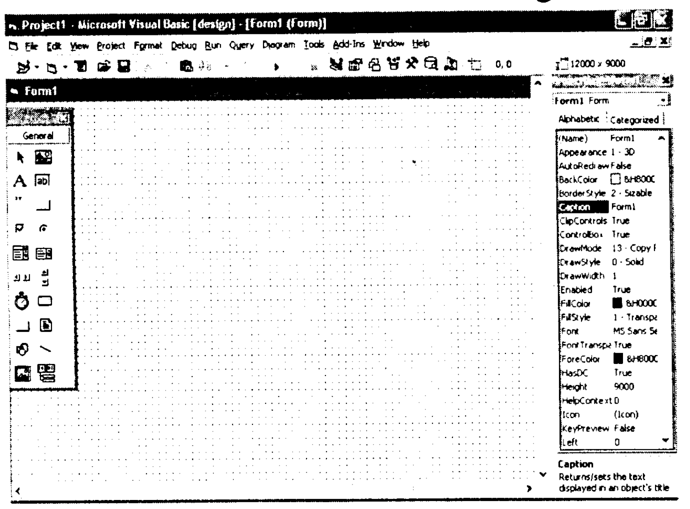
والنوافذ المعروضة هي ما يلي:

أ – نافذة أدوات Tools:



تجدها علي يسار الشاشة. وبها مجموعة متنوعة من الكائنات الـتي تسـتخدم لتكـوين البرنامج، حيث يوجد شكل خاص لكتابة البيانات بنافذة البرنامج، وآخر لإضافة الصور إلى النافذة. والكائنات هي العناصر الأساسية التي ستستخدم لتكوين البرنامج لاحقاً.

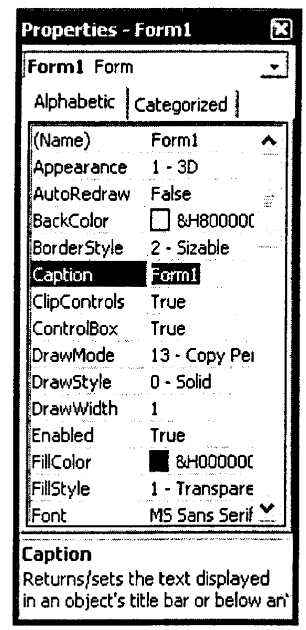
ب – نافذة نموذج From



تستخدم هذه النافذة لتصميم واجهة البرنامج الذي تصممه، فعليها سـتكون العنـاوين وصناديق إدخال البيانات أو صناديق اختيار البيانات، وازرار الأوامر بالإضافة إلى عرض الصور، وهى المكان الأساسي الذي يستخدم لبدء تصميم البرنامج.

وكلما تطلب الأمر يمكنك استخدام نافذة نموذج أخري إلى جانب هذه النافذة.

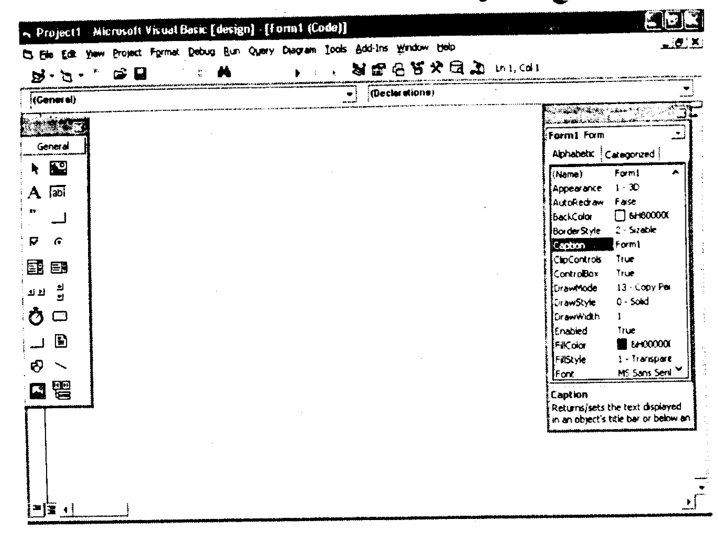
ج- نافذة خصائص Properties:



تستخدم هذه النافذة لتحديد صفات الشكل Object والذي قد يكون صندوق نص كتابة، أو اختيار، أو أمر، أو صندوق لعرض أو استقبال البيانات.

وتشتمل خصائص المشروع Project على تحديد صفة الشكل مثل اللون – الخط – الحجم – العنوان الخ....... ويمكن تحديد ٤٤ صفة لكل شكل.

د – نافذة مشروع **Project**:



يمثل العناصر التي تستخدم لتكوين البرنامج، بالإضافة إلى الوحدات التي تضـاف لكـي يعمل البرنامج بطريقة سليمة، وهذه الوحدات تضاف تلقائياً بواسطة اللغة، ومن هـذه النافذة سنكتب الأوامر المرتبطة مع الكائنات التي ستكون بنافذة التصميم والتي اسمهـا **From1**.

هـ– نافذة تصميم **Design**:

ستجد هذه النافذة في اعلي الشاشة، وبها اسم البرنامج – وأسماء القوائم – والأيقونات.

كائنات صندوق الأدوات **Toolbox**:

بصندوق الأدوات مجموعة من الكائنات التي يختار المبرمج منها بناء علي الوظيفـة الـتي سيستخدمها، ويساعد معرفة هذه الكائنات في كتابة البرنامج، وفيمـا يلـي توضـيح للكائنات التي توجد بالإصدار العادي، وهى الأكثر استخداماً.

صندوق الصور Picture Box:

الاسم المفترض الذي تستخدمه اللغة Picture1، ورقم ١ يعني أن هنالك شكلاً واحداً حتى الآن، وإذا استخدمت مزيد من هذه الصناديق داخل البرنامج فان الأرقام ستزيد رقماً كل مرة عند إضافة شكل جديد.

ويستخدم هذا الشكل لعرض الصور داخل الإطار المرسوم أثناء التنفيذ مثل صورة الموظف، أو صورة الطالب، أو صورة قطعة الغيار، أو صورة السلعة. والصور التي يمكن عرضها داخل هذا الشكل لها الامتداد Bmp. أو Wmf. أو Dip.

عنوان Label:

الاسم المفترض Label1 ويستخدم هذا الكائن لتكوين عناوين بالنموذج يوضح نوع البيانات أو عمل البرامج، وتحدد البيانات بالعناوين بواسطة المبرمج.

صندوق النص Text Box:

الاسم المفترض له Textl ويستخدم هذا الشكل لتكوين مكان بالشاشة يستخدم لاحقاً لإدخال البيانات عند تنفيذ البرنامج، وهذه البيانات قد تكون حرفية أو رقمية.

إطار Frame:

الاسم المفترض Frame1 ويستخدم لتجميع مجموعة من الكائنات بإطار واحد مما يساعد علي نقلها معاً عند الرغبة في تعديل أماكنها لاحقاً.

أزرار الأوامر Command Button:

الاسم المفترض command1. ويستخدم هذا الكائن لتكوين أزرار الأوامر مثل اختيار موافق OK أو إلغاء الأمر Cancel. وتستجيب هذه الاختيارات للمستخدم عند الضغط عليها بالفارة أو باستخدام مفاتيح الاختصارات بالضغط علي حرف معين يحدد مسبقاً مع مفتاح Alt بدلاً من استخدام الفارة إذا أردنا ذلك.

صندوق الاختيار Check Box:

الاسم المفترض Check1. ويستخدم هذا الكائن لتكوين افتراضات معينة أثناء تنفيذ البرنامج، ويحدد المستخدم لاحقاً هذه الافتراضات مثل تغيير شكل الخط، تحديد نوع الطباعة.... الخ، وتحدد هذه الافتراضات بوضع حرف X داخل الإطار المربع الذي يظهر إلى جانب الاسم المفترض ويمكن اختيار أكثر من صندوق.

دائرة الاختيار Option Button:

الاسم المفترض Option1. ولا تختلف وظيفة هذا الشكل عن وظيفة الشكل السابق سوى أن المستخدم لا يستطيع سوى تحديد اختيار واحد من مجموعة اختيارات.

قائمة البيانات ComboBox:

الاسم المفترض Combo1، ويستخدم لتكوين قائمة البيانات والتي تشبه بعملها قوائم ويندوز حيث علي المستخدم اختيار أحد سطور البيانات منها، وبهذه القائمة عموداً للتحريك لأسفل ولأعلي إذا لم تكن هناك مساحة كافية لعرض كل البيانات، وذلك لاستعراض بقية البيانات. والبيانات التي توضع بهذه القائمة من النوع الذي لا يتبدل أثناء عمل البرنامج، فعند إعداد برنامج لتكوين العناوين، يمكن استخدام هذا الشكل لحفظ أسماء المدن، بدلاً من كتابة الاسم يدوياً، ويستطيع المستخدم اختيار الاسم من القائمة المعدة مسبقاً، وإذا لم يكن الاسم المطلوب داخل هذه القائمة فيمكن كتابة الاسم يدوياً، ويستطيع المستخدم اختيار احدي البيانات من القائمة ولا يستطيع اختيار عدة بيانات في نفس الوقت، ولا يحتل هذا النوع من الكائنات مساحة كبيرة علي الشاشة، حيث تتحرك البيانات عمودياً بالقائمة.

قائمة الاختيارات List Box:

الاسم المفترض List1. وهذا الكائن مشابه في العمل مع الكائن السابق مع فارق وحيد هو انه علي المستخدم اختيار أحد البيانات من القائمة ولا يستطيع كتابة بيانات أخري إذا لم تكن موجودة بالقائمة. ويمكن تحديد عدة اختيارات من القائمة التي تظهر أمامه، والتي

تتطلب مساحة محدودة وكافية علي الشاشة لعرض البيانات المحددة مسبقاً. مثل الحالة الدراسية (ناجح – منقول بمواد – راسب – غائب).

عمود التحريك أفقياً **Horizontal Scroll Bar**:

الاسم المفترض **Hscroll**. ويستخدم لتكوين عمود يحتوى علي قيم معينة تتبدل أثناء تحريك مؤشر الأسهم.

عمود التحريك الراسي **Vertical Scroll Bar**:

الاسم المفترض **Vscroll**. ويستخدم لتكوين عمود يحتوى علي قيم معينة تتبدل أثناء تحريك مؤشر الأسهم.

الميقاتى **Timer**:

الاسم المفترض **Timer1**. يستخدم لإضافة ميقاتي زمني بالبرنامج، وتحدد لهذا الميقاتى زمني البدء والانتهاء من قبل البرنامج، ولا تعمل هذه الوظيفة إلا عندما يحين الوقت المحدد مسبقاً، حيث يتولى الميقاتى مراقبة الوقت إلى أن يتطابق مع وقت تشغيل الوظيفة فيقوم بتنفيذها، وهذا الميقاتى مفيد للبرامج التي يجب أن تعمل بعد فترة معينة، مثل برامج الاتصال أو النسخ الاحتياطي أو تحريك الأشكال والصور.

قائمة أسماء الاسطوانات **List Drive Box**:

الاسم المفترض **Drive1**. وتستخدم لعرض أسماء وحدات الاسطوانات المرتبطة بالجهاز، وهذا الكائن مفيد بالبرامج التي تتعامل مع الملفات، حيث نستطيع تحديد الاسطوانة المطلوبة لفتح أو حفظ الملفات.

قائمة أسماء الفهارس **Directory Box**:

الاسم المفترض **Dir1**. يستخدم هذا الشكل لإظهار أسماء الفهارس الموجودة بالاسطوانة المختارة، مما يساعد علي تحديد أماكن الملفات.

قائمة أسماء الملفات File List Box:

يستخدم هذا الشكل لعرض أسماء الملفات بالاسطوانة أو الفهرس الحالي، وذلك لاختيار اسم ملف منها.

قالب/ شكل Shape

الاسم المفترض Shape1. يستخدم هذا الكائن لرسم كائنات هندسية مختلفة مربع، دائرة....

خط Line:

الاسم المفترض له Line1. ويستخدم لرسم خطوط بالشاشة.

عرض الصورة Image:

الاسم المفترض له Image1. ويستخدم لتحديد مكان لعرض الصورة لاحقاً.

بيانات Data:

الاسم المفترض له Data1. ويستخدم للتعامل مع ملفات قواعد البيانات المصممة ببرامج أخري مثل اكسس Access.

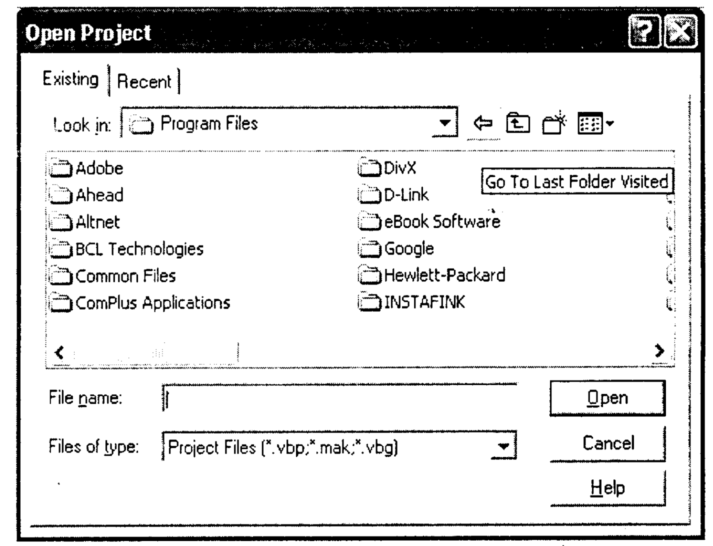
صناديق الحوار Common Dialog Boxes:

من مميزات النوافذ أنها تستخدم عناصر متعارف عليها لإدخال البيانات من المستخدم ولإخراج النتائج له مثل قوائم الاختيار وصناديق السرد وتتطلب بعض التطبيقات البحث عن اسم ملف، أو إعداد آلة الطباعة لطباعة عدة صفحات من ملف، أو اختيار احد الابناط المتاحة وكل ذلك تقدمه لنا لغة البيسك المرئي بواسطة صناديق الحوار القياسية وهى تتألف من ستة صناديق يتم اختيار إحداها أثناء تنفيذ البرنامج عن طريق وضع نسخة من عنصر صندوق الحوار Common Dialog ثم تغيير خاصية Action تبعاً للجدول التالي والذي يوضح صندوق الحوار وقيمة خاصية Action التي تظهره.

صندوق الحوار	قيمة خاصية **Action**
صندوق فتح الملفات	١
صندوق حفظ الملفات	٢
صندوق الألوان	٣
صندوق الابناط	٤
صندوق الطباعة	٥
شاشة المساعدة	٦

وما يلي شرح لهذه الصناديق:

١- صندوق فتح الملفات **Open**:



يستخدم لاستدعاء ملف لاستخراج البيانات منه وتنتهي وظيفة هذا الصندوق بمجـرد اختيار اسم ملف من قائمة الملفات والتحقق من وجوده فإذا حاول المستخدم كتابة اسم ملف غير موجود علي الاسطوانة أو خطأ في أحد الأدلة الفرعية، ستعرض رسالة حدوث خطأ من صندوق الفتح ويتم تحميل هذا الصندوق بمجرد أن يكتب المبرمج:

Cmd1.action = 1

حيث Cmd1 هو اسم العنصر من هذا النوع. ويمكن للمستخدم التنقل بـين الأدلـة المختلفة وبين الاسطوانات المختلفة الموجودة في الجهاز وبعد الضغط علي مفتاح Enter أو النقر علي Ok يحتفظ الصندوق باسم الملـف المختـار في المـتغيرين Filetitle و Filename ويحوى Filetitle اسم الملف فقط بينما يحوى Filename المسـار الكامل للملف فإذا أردنا استخدام الملف بعد ذلك في عملية فتح حقيقية يجب استخدام filename ويمكننا تغيير عنوان الصندوق بكتابة

Cmd1.dialogtitle = Filename

ويمكنك تحديد امتدادات معينة للملفات وذلك باستخدام خاصية filter ويمكننا وضـع عدة امتدادات للاختيار منها ويفصل بين كل منها علامة كما يمكننا التعامل مع حروف الإبدال Wild Cards كما يوضح المثال التالي:

D1.filter = "Text |*.txt|Data| *.dat"

والذي يعنى وجود امتدادين txt. و dat. ويبدأ الصندوق وهو يعرض الأول وبالتـالي ينتقي الملفات ذات الامتداد الأول ويمكن للمستخدم الانتقال بين الامتدادات المختلفـة عن طريق صندوق السرد Combo box في اسفل يسار صندوق فتح الملفات.

٢- صندوق حفظ الملفات Save As:

Save File As

Save in: Microsoft Visual Studio

Common
VB98
VC98
Vfp98
VIntDev98

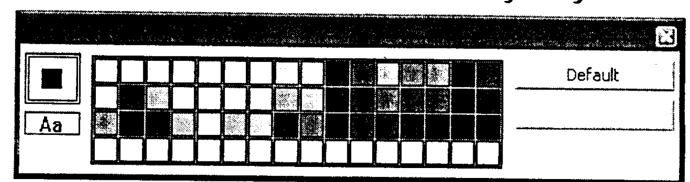
File name: Form1
Save as type: Form Files (*.frm)

Save
Cancel
Help

وهو يشبه صندوق فتح الملفات وسندرس كيفية التصرف إذا رغبنا في عدم استكمال الحفظ لأي سبب فسنضغط علي **Cancel**

حتى يدرك البرنامج طريقة خروجنا من صندوق حوار الحفظ و ليقرر ما إذا كان سيقوم بحفظ الملف من عدمه يتم إحداث خطأ من نوع خاص هو **CDERR_CANCEL** عند ضغط زرار **Cancel** ويجب تصيد هذا الخطأ حتى لا يتم الحفظ عند ضغط زرار **Cancel** ويتضح من الشكل السابق اختيار الحفظ.

٣- صندوق الألوان Colors

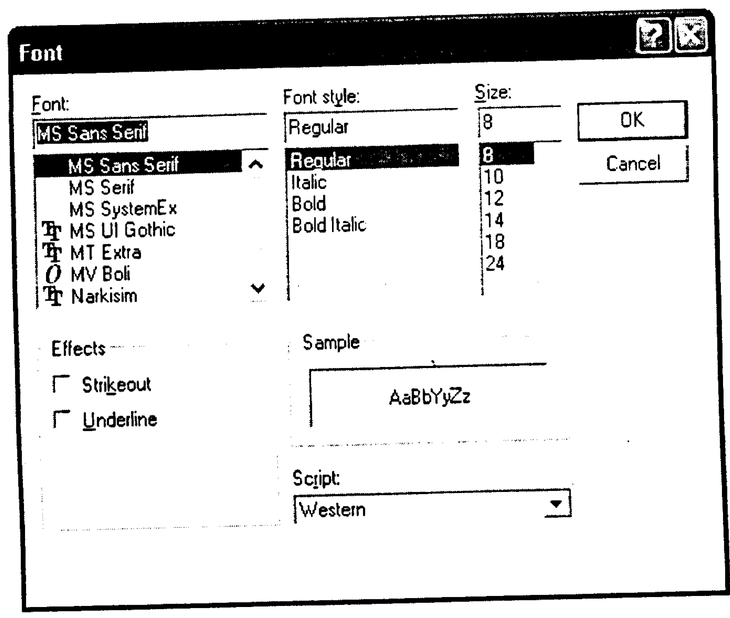


يوضح الشكل السابق صندوق الألوان. لاختيار احد الألوان انقرها نقرة مزدوجة أو نقرة واحدة ثم ضغط زرار **Enter** واللون الناتج سيخزن في خاصية **Color** فمثلاً لجعل لون الخلفية اللون المختار اكتب السطرين التاليين:

```
Cmd.action = 4
Form1.backcolor = Cmd1.color
```

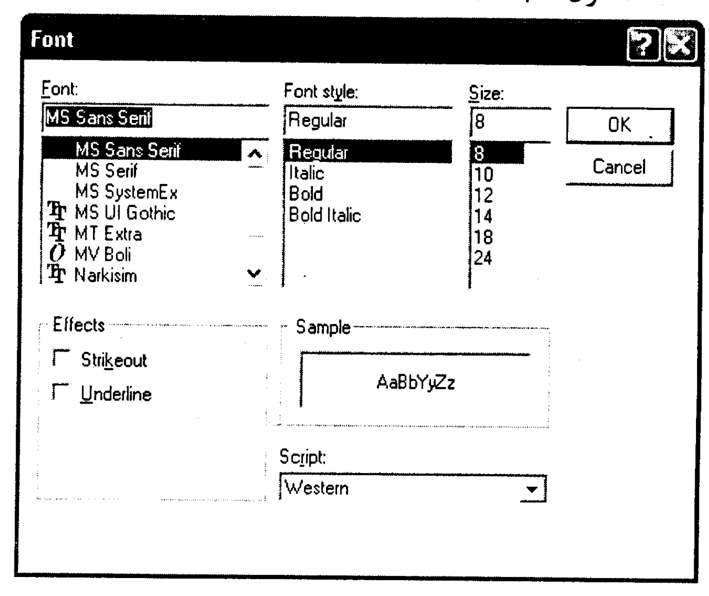
أما إذا أردت مزج ألواناً بنفسك اضغط زرار التحكم **Define Custom Color** فيزداد الصندوق عرضاً ويمكنك تحريك المؤشر الموجود فوق المنطقة متعددة الألوان عن طريق سحبه بالفارة حتى تقف الفارة على لونك المختار ثم تضغط مفتاح إدخال **Enter** لقبول اللون المحدد.

٤- صندوق الابناط Fonts:

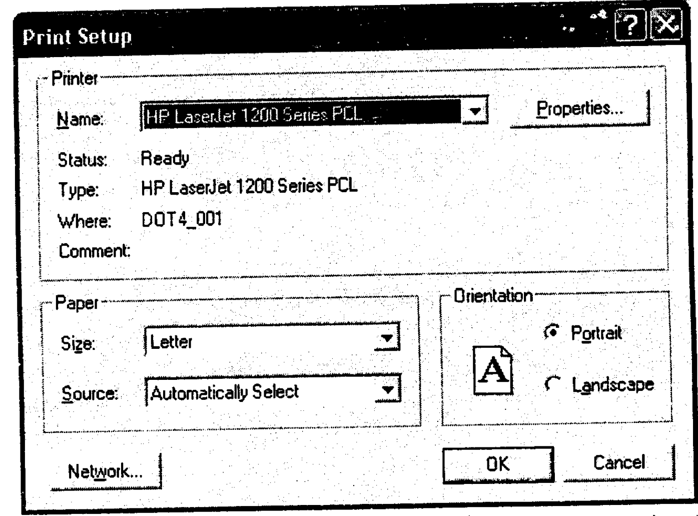


يمكنك تحديد بنط من الابناط المتاحة وتحديد حجمه باستخدام خاصية **Fontsize** ويجب إلا يزيد الحجم عن **Max** وألا يقل عن **Min** وكذلك من مخرجات الصندوق **FontItalic** لتحديد ميل الخط من عدمه وكذا **Fontbold** لتثقيل الخط و **FontUnderline** لوضع خط تحت الحرف أما تحديد البنط نفسه فيتم باستخدام خاصية **FontName**.

٥- صندوق الطباعة Print:



يمكنك تحديد عدد نسخ الطباعة وأول وآخر صفحة وأشياء أخرى عن الطباعة. بجعل خاصية Action تساوى ٥ سيعرض صندوق الطباعة إما All كل التقرير أو Current Page الصفحة الحالية أو Pages من صفحة إلي صفحة، وعدد النسخ في خاصية Copies. ويوضح الشكل السابق صندوق الطباعة وإذا نقرت علي زرار Setup فيعرض صندوق إعداد الطابعة.



يمكنك إظهار شاشة المساعدة لأحد ملفات المساعدة **Help**.* بجعل خاصية **Action** تساوى ٦ مع تحديد ملف المساعدة الخاص ببرنامجك كما يلي:

App.helpfile = "c:\vb\vb.help"

المرئي كلما جعلت خاصية **Action** تساوى ٦ أو ضغطت علي **F1**.

الفصل الرابع

تحديد خصائص الكائن

Properties

Properties - Form1

Form1 Form

Alphabetic | Categorized

(Name)	Form1
Appearance	1 - 3D
AutoRedraw	False
BackColor	&H80000C
BorderStyle	2 - Sizable
Caption	Form1
ClipControls	True
ControlBox	True
DrawMode	13 - Copy Pe
DrawStyle	0 - Solid
DrawWidth	1
Enabled	True
FillColor	&H00000C
FillStyle	1 - Transpare
Font	MS Sans Serif

Caption
Returns/sets the text displayed in an object's title bar or below an

بعد إضافة الكائن أثناء تصميم شاشة البرنامج مثل الزرار أو صندوق النص أو صندوق الصور يجب تحديد خصائص هذا الكائن، والتي تساعد علي تحديد هيئة الكائن وعمله أثناء تنفيذ البرنامج، وبغض الخصائص يجب تحديدها أثناء تصميم البرنامج والبعض الآخر يمكن تعديله أو تحديده أثناء تنفيذ البرنامج بكتابة الأوامر المناسبة لذلك. والخصائص التي يمكن تحديدها لكائن معين متعددة وكثيرة، لهذا سنعرض الخصائص التي يمكن تعديلها أثناء التصميم، والخصائص التالية ستجدها أثناء التصميم، وتظهر بناء علي نوع الكائن، اى أن أسماء الخصائص التالية لن تجدها ثابتة لكل كائن، بل تختلف من كائن إلي آخر، ولذلك تم دمج خصائص الأشكال بجدول لاشتراك بعض الأشكال في الخصائص المتعددة.

ولكل كائن ما يقرب من ٤٠ صفة خاصة وعامة. ويوجد ما يزيد علي ٧٠ من هذه الخصائص المتاحة لكل الأشكال أثناء التصميم. ولقد ذكرنا تقريباً لأن بعض الأشكال لا تظهر إلا عند استخدام الإصدار الخاص بالمحترفين، وسنعرض فيما يلي خصائص الأشكال الخاصة بالإصدار العادي.

ومن المهم الإلمام بهذه الخصائص لأنها تساعدك في تصميم البرنامج بكائن واضح، ولكون البرمجة باستخدام لغة فيجوال بيسك تعتمد علي الكائن في معظمها، فإن معرفة الخصائص أساسية لمتابعة البرمجة، وتعدد الخصائص لا يعنى بالضرورة استخدامها جميعاً، لأن تحديد الصفة لكائن معين يرتبط ارتباطاً وثيقاً بنوع الوظيفة التي ستحدد للكائن.

وما يلي الخصائص الأساسية للأشكال:

Align: لتحديد مكان وطريقة عرض الصورة بالشاشة أثناء تنفيذ البرنامج.
Alignment: لتحديد طريقة عرض النص من اليسار – الوسط – اليمين.
AutoRedraw: لإعادة رسم الصورة أو الكائن مرة ثانية بصورة تلقائية أو يدوية بأمر معين.
Autosize: لتحديد مقاس الكائن بناء علي حجم البيانات التي تعرض داخله، هذه الميزة مفيدة عند عدم معرفة حجم البيانات أو النص الذي سيعرض داخل الكائن.
Backcolor: لتحديد اللون الخلفي للكائن.
BorderStyle: لتحديد نوع أطار الكائن.
BorderWidth: لتحديد سمك الإطار.
Cancel: لتحديد إذا كان الزرار يمكن استخدامه أم انه ملغي مؤقتاً.
Caption: لكتابة النص داخل مربع النص أو الزرار، ليظهر هذا النص علي الشاشة.
ClipControls: لتحديد نوع إعادة الرسم بالكائن مرة ثانية، حيث يوجد تحديد لإعادة الرسم بكائن كلي أو بالجزء المستخدم.
ControlBox: لإخفاء أو إظهار القوائم من علي شاشة البرنامج.

DatabaseName: لتحديد اسم ومكان ملف البيانات الذي سنستخدمه.

DataField: لتحديد اسم الحقل المطلوب الوصول إليه بالملف المستخدم.

DataSource: لتحديد السيطرة علي الملف المستخدم.

Default: لتحديد زرار الأوامر الافتراضي مسبقاً، تعرض بعض الشاشات مجموعة من الأزرار ونجد دائماً زراراً افتراضياً بهذه الشاشة، مثلاً عند إلغاء احد الملفات، سنجد أن زراراً لا هو المفترض من بين الأزرار.

DragIcon: لتحديد الأيقونة التي ستظهر عند تصغير شاشة البرنامج.

DragMode: لتحديد نوع رسم الكائن.

DrawMode: لتحديد أسلوب خط الرسم.

DrawStyle: لتحديد نمط خط الرسم.

DrawWidth: لتحديد سمك خط الرسم.

Enabled: لتحديد إذا كان بالاستطاعة استخدام الكائن أم لا، ويعتمد استخدام الكائن علي إحداث تأثير بواسطة الفارة أو لوحة المفاتيح.

Exclusive: لتحديد صفة استخدام ملف قواعد البيانات، وهذه الصفة إما تكون مشتركة أو مفردة.

FillColor: لتحديد اللون الذي سيصبغ به الكائن.

FileStyle: لتحديد حاشية الرسم.

FontBold: استخدام خط غامق للكتابة.

FontItalic: استخدام خط مائل للكتابة.

FontName: تحديد اسم الخط الذي سيستخدم.

FontSize: لتحديد قياس الخط.

FontStrikeThrought: لكتابة اسطر بمنتصف الحروف.

FontTransparent: لتحديد خط شفاف للكتابة.

FontUnderline: لوضع خط تحت النص.

ForeColor: لتحديد اللون الامامي.

Hight: لتحديد ارتفاع الكائن.

HelpContextid ملف مساعدة يحتوى علي عدة أجزاء وكل جزء يشرح وظيفة معينة. هذه الخاصية تساعد علي استدعاء ذلك الجزء عند الضرورة باستخدام مفتاح **F1** عند تنفيذ البرنامج، حيث يتم تحديد جزء المساعدة برقم معين. وترتبط المساعدة المعروضة بالكائن المستخدم.

Icon: لتحديد الأيقونة المناسبة بعد تصغير شاشة البرنامج. وهى تستخدم لإعادة شاشة البرنامج إلى الوضع السابق، وتوجد أيقونة مفترضة. يمكن تغييرها إذا أردت بتحديد أيقونة أخري من فهرس الأيقونات الذي يوجد بفهرس لغة فيجوال بيسك.

Index: لتحديد أرقام المجموعات، والمجموعات عبارة عن أشكال من فئة واحدة، مثل صناديق النص، أو أزرار الاختيار. وترقيم أشكال المجموعة يساعد في العامل معها.

Interval: لتحديد الفارق الزمني الذي سيستخدم لقياس الوقت لتنفيذ مهمة معينة.

KeyPreview: لعرض نص بوظيفة المفاتيح مثل **F1** أو **F8** قبل أن تبدأ هذه المفاتيح بعملها، إذا ضغطت علي مفتاح **F1** بغرض أن هذا المفتاح مخصص للمساعدة سيعرض اسم هذا المفتاح **Help**، فإذا أراد المستخدم طلب ملف المساعدة فعليا فعليه الضغط علي هذا المفتاح مرة ثانية، مما يساعد علي التأكد من استعمال المفتاح الصحيح.

Left: لتحديد المسافة من اليسار بين شاشة البرنامج وشاشة الحاسب.

LinkMode: لتحديد نوع العلاقة مع التطبيقات الاخري.

LinkItem: لتحديد البيانات التي ترتبط مع التطبيقات الاخري.

LinkTimeout: لتحديد الفترة الزمنية المطلوبة لاستجابة الربط مع التطبيقات الاخري.

LinkTopic: لتحديد مصدر الربط.

MaxButton: لتحديد إذا كان باستطاعة المستخدم استعمال صندوق التكبير الخاص بشاشة البرنامج.

MdicChild: ربط البيانات بين تطبيقات ويندوز ينشأ عنها ظهور شاشات جديدة، هذه الشاشات قد تحدد للظهور في مكان واحد حيث تسمى الشاشة التي تستدعى مكان الشاشة السابقة باسم الشاشة الابن. فعند استخدام برنامج وورد مثلاً فإن كل الشاشات المرتبطة بهذا البرنامج تظهر بمكان جانبي، هذه الخاصية مفيدة لمتابعة برنامج معين بدقة.

MinButton: لتحديد إذا كان باستطاعة المستخدم استعمال صـــندوق التصـــغير الخاص بشاشة البرنامج.

MousePointer : لتغيير كائن مؤشر الفارة أثناء تنفيذ البرنامج، وتوجـــد عـــدة أشكال يمكن استبدالها بدلاً من المؤشر العادي.

MultiLine: إذا استخدمت صندوق النص أو القائمة، قد لا تستطيع معرفة البيانات التي ستستخدم، هذه الميزة تسمح باستخدام عدة سطور تندرج بالصندوق.

Name: يمثل اسم الكائن أهمية بالنسبة للمبرمج أثناء كتابة البرنامج، فعند تكوين كائن لاستقبال اسم المدينة مثلاً، فإن الكائن سيأخذ الاسم التالي .Text1 وهـــذا لا يعطـــى مرونة في كتابة الأوامر، وبدلاً من ذلك يمكن إعادة استدعاء تسمية الكائن من .Text1 إلي اسم آخر مثل City، ويمكن إعادة تسمية معظم الأشكال إلى أسماء يحددها المبرمج.

PasswordCher: لإخفاء حروف كلمة السر أثناء كتابتها هذه الخاصية مفيدة إذا استخدمت كلمة السر للوصول إلى بيانات معينة ولا ترغب في أن يراهـــا أحـــد أثنـــاء كتابتها.

Picture: لكتابة اسم ملف الصورة التي ستظهر داخل إطار كائن الصورة.

ReadOnly: للسيطرة علي التسجيل عند استخدام ملف قواعد البيانات.

ScaleHeight: لتحديد المساحة التي ستستخدم بالكائن، وهى خاصة بالصور.

ScaleMode: لتحديد وحدة قياس الصورة مثل الوحدة أو النقطة.

ScaleTop: لتحديد الزاوية العليا من الكائن الخاص بالصورة، والتي تستخدم لاحقاً عند عرض الصورة بالكائن بدءاً من هذه الزاوية.

ScrollBars: لعرض أعمدة التحريك بالكائن، هذه الصفة مفيدة عند عرض البيانات أو الصور بالأشكال، لأن البيانات في بعض الأحيان قد لا تظهر بالكائن الكامل.

Shape: لتحديد نوع الرسم الذي سيظهر لك مثل دائرة أو مستطيل أو قوس.

Sorted: إذا كنت تستخدم قائمة البيانات، فإن هذه الصفة تعمل علي ترتيب البيانات الموجودة داخل القائمة تلقائياً.

Stretch: إذا كنت تستخدم صورة ضمن إطار فإن هذه الصفة تعمل علـــي ترتيـــب البيانات الموجودة داخل القائمة تلقائياً.

Style: لتحديد طريقة عمل صندوق القوائم، حيث يمكن للمستخدم اختيار البيانـات من القائمة أو تحديد إذا كان باستطاعة المستخدم كافة البيانات يدوياً، وهي تشبه فـتح الملفات، حيث يمكن للمستخدم اختيار اسم الملف من قائمة الملفات أو كتابته يدوياً.

TabIndex: لتحديد أولوية الانتقال بين الأشكال، حيث يمكن تحديد انتقال المؤشر من كائن إلي آخر بناء علي تحديد رقم كل كائن، وينتقل المؤشر بناء علي أولوية أرقـام هذه الأشكال مما يساعد علي تنظيم إدخال النص أو اختيار الأوامر أثناء التنفيذ.

Tag: رمز يستخدم لمراقبة استعمال كائن محدد، وبناء علي هذا يحدد لاحقـاً طريقـة التعامل مع الكائن بناءاً علي إشارة الرمز، إذا كان هنالك صورة مطلوب تحريكها مـن مكان إلي آخر، بعد تحريك الصورة تتغير الإشارة المحددة لهذه الصورة، لهذا قد يتم كتابة أمر لعدم تحريك الصورة من موضعها الجديد بناء علي نوع الإشارة التي تغيرت.

TapStop: لتحديد إذا كان باستطاعة المستخدم استعمال مفتاح **Tap** للانتقال بين صناديق الأشكال أو لا، الصناديق مثل الأزرار أو صناديق إدخال البيانات، وكل كـائن يحدد مسبقاً لهذه الغاية.

Text: لتحديد البيانات الافتراضية بصندوق البيانات أو قوائم البيانات.

Top: لتحديد المسافة بين اعلي الكائن واعلي المكان الموضوع به.

Value: لتحديد افتراض مسبق لصناديق الاختيار، مثلاً عند طباعة البيانات سـتجد صندوقاً افتراضياً لطبع كل الصفحات، يمكن تعديله باختيار تحديد آخر لطبـع صـفحة معينة بدلاً من التحديد الحالي.**Visible**: لتحديد إذا كان الكائن مرئيـاً أم لا، هـذه الصفحة مفيدة لإظهار الأشكال وإخفائها عند تعددها علي الشاشة، حيث يمكن وضـع كائن جديد مكان الكائن السابق، مما يمكنك من استخدام شاشة البرنامج بطريقة أفضل.

Width: لتحديد عرض الكائن أو الشاشة.

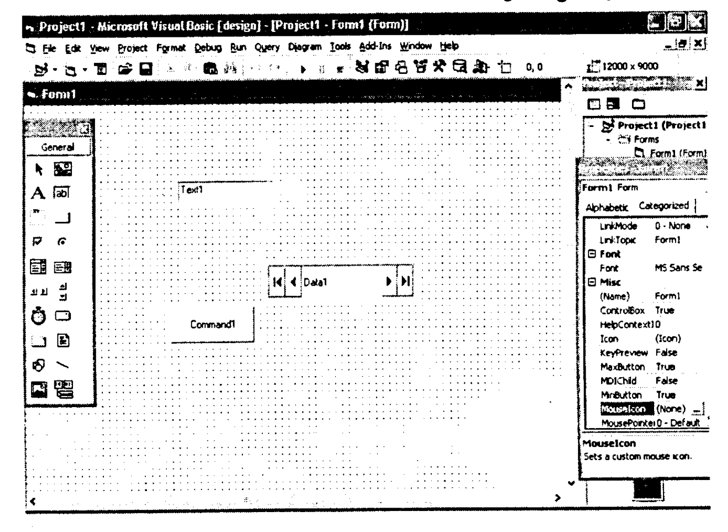
WindowState: لتحديد عرض شاشة البرنامج، حيـث تصـغيرها أو تكبيرهـا باستخدام هذه الصفة، ويمكن تعديلها بأوامر من داخل البرنامج، وهـي مفيـدة عنـد استخدام عدة شاشات أثناء عمل البرنامج، حيث تصغر الشاشة في أيقونـة ثم تعـاد إلى حالتها السابقة باستخدام الفارة أو بأمر بالبرنامج.

تغيير خصائص الأشكال:

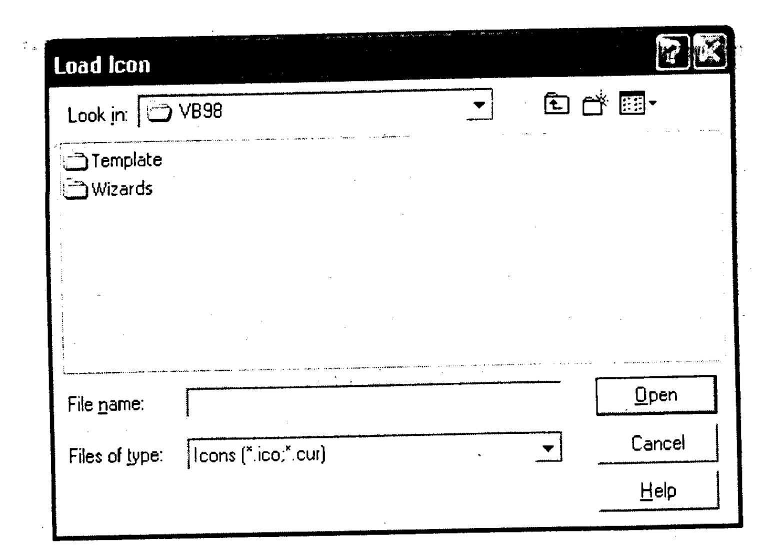
الأشكال عبارة عن إطارات ترسم بواجهة البرنامج **From** لأداء غرض معين، وكل كائن له إطاره ووظيفته الخاصة. ويمكن تغيير خصائص الإطار الخاص بالكائن بشاشة التصميم أثناء إعداد واجهة البرنامج، حيث يمكن استخدام نافذة الخصائص لتحديد الخصائص الجديدة للكائن. وعند تحديد نافذة الخصائص لتحديد الخصائص الجديدة للكائن. وعند تحديد الكائن المطلوب تظهر لك خصائصه بنافذة الخصائص، وستلاحظ هذا التغيير بنافذة الخصائص **Properties**. يمكن تعديل الخصائص بأوامر البرنامج حيث قد نحتاج لذلك لأسباب عديدة وفيما يلي طرق تعديل خصائص الكائن.

تعديل خصائص الكائن بشاشة التصميم:

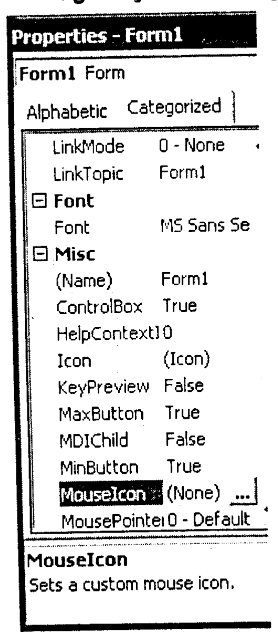
١- حدد إطار الكل المطلوب.



٢- ضع الفأرة بنافذة الخصائص ثم اختار الصفة المطلوبة.



٣- اكتب أو حدد الخصائص الجديدة بالسطر الخاص بالخصائص المطلوبة.



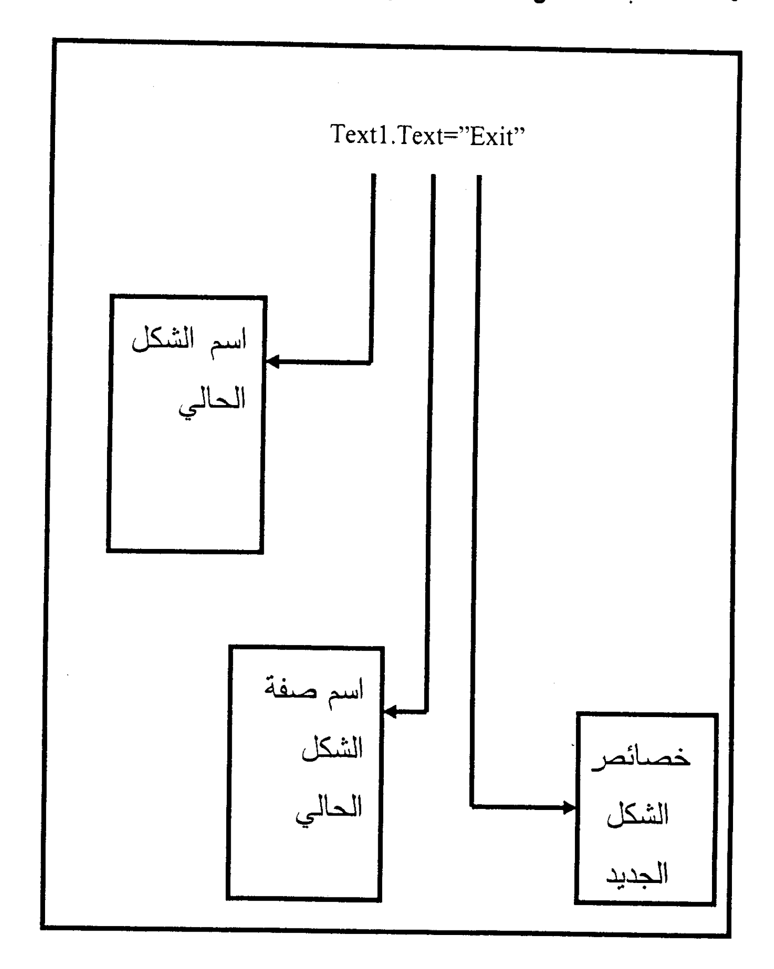
تعديل الخصائص بالأوامر:

١- اكتب اسم الكائن.

٢- ضع نقطة.

٣- . اكتب اسم الصفة.

٤- اكتب الخصائص الجديدة بعد إشارة التساوي. كما يظهر بالكائن التالي:



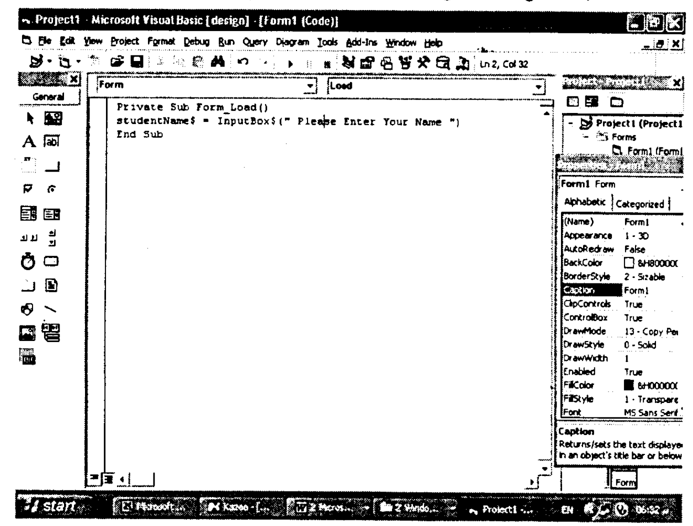
المتغيرات والثوابت والإجراءات في لغة فيجوال بيسك

تحتاج برامج لغات الحاسب إلى التفرقة بين المتغيرات والثوابت التي ستستخدم بالبرنامج.

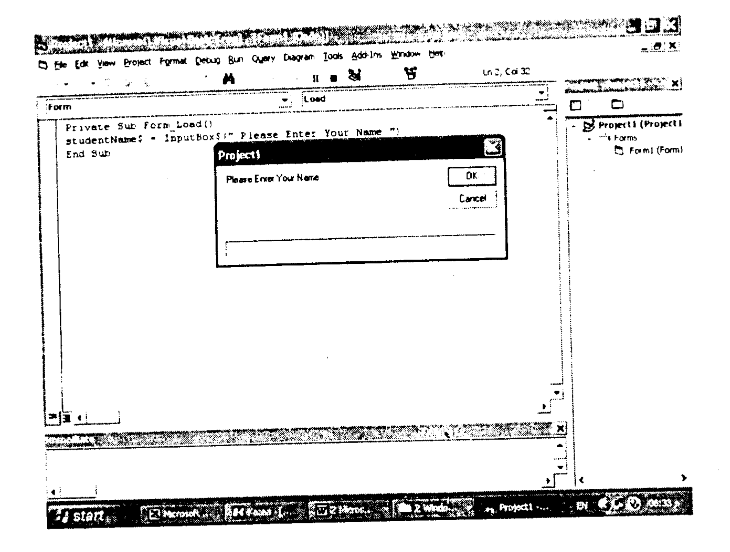
المتغيرات Variables:

المتغير هو قيمة تتغير عند تشغيل البرنامج، وذلك مثل أسم الطالب أو أسم السلعة حيث يتغير البيان من سجل لآخر. لذلك نستخدم له متغير، ولـيكن StudentName$ ويمكنك كتابة أمر:

StudentName$ = InputBox$ ("ادخل اسمك من فضلك ")



سيعرض "فيجوال بيسك" صندوق الحوار التالي:



يستخدم أمر InputBox$ لطلب معلومة من المستخدم فيمكنك فيجوال بيسك من إدخال اسم الطالب في الخانة الموجودة في أسفل الصندوق. وعندما ينتهي المستخدم من إدخال الاسم، يتم وضع هذا الاسم في المتغير StudentName$. ويحتفظ المتغير بهذه القيمة إلى أن تقوم بتغييرها.

ويمكنك استخدام القيمة التي يحويها المتغير بوضعها في متغير آخر، مثلاً:

UserName$ = StudentName$

فيتم نسخ القيمة الموجودة في المتغير StudentName$ إلى المتغير UserName$.

ويمكن وضع هذه القيمة كعنصر في قائمة مثلاً:

List1.Additem = studentName$

ويمكنك تغيير قيمة المتغير بوضع قيمة جديدة داخله، فيلغى فيجوال بيسك القيمة القديمة ويضع محلها القيمة الجديدة. مثلاً:

StudentName$ = "عمرو محمد"

وهذا المتغير يسمى متغير حرفي String Variable لأنه يحتوى علي مجموعة حروف. وتوجد متغيرات عددية تستخدم للإعداد. فمثلاً يمكنك تحديد متغير باسم سعر الوحدة UnitPrice وتضع القيمة في متغير عددي:

UnitPrice = InputBox$ "سعر الوحدة"

ويتميز المتغير العددي بإمكانية إجراء العمليات الحسابية عليه مثل الطرح والضرب والقسمة.

قواعد تحديد اسم المتغير:

- يجب ألا يزيد اسم المتغير عن ٤٠ حرفاً.
- يجب أن يكون أول حرف منه حرفاً أبجديا.
- يجب ألا تستخدم كلمة من الكلمات المحجوزة للغة Reserved Word. والكلمات المحجوزة هي مجموعة الأوامر والعبارات التي يستخدمها فيجوال بيسك فلا يمكن استخدام كلمة مثل For كأسم لمتغير، ولكن يمكن استخدامها كجزء من اسم المتغير مثل ForPrint.

ويفضل تجنب تسمية المتغيرات بأسماء الأدوات التي تضعها علي النافذة أو بأسماء الخصائص، لان ذلك وإن لم يكن ممنوعاً إلا أنه قد يتسبب في الخلط بينها.

أنواع المتغيرات:

هناك عدة أنواع من المتغيرات في فيجوال بيسك، كما يوضح الجدول التالي:

أنواع المتغيرات:

نوع المتغير	معناه	مداه
Integer	عدد صحيح (٢ بايت)	من ٣٢٧٦٨ – إلى ٣٢٧٦٧
Long	عدد صحيح (٤ بايت)	من ٢١٤٧٤٨٣٦٤٨ – إلى ٢١٤٧٤٨٣٦٧٤
Single	عدد ذو فاصلة عشرية عائمة **Floating Point** (٤ بايت)	من ٣,٤٠٢٨٢٣**E38** – إلى **E-45** ١,٤٠١٢٩٨ إلي **E38** ٣,٤٠٢٨٢٣ (قيم موجبة)
Double	عدد ذو فاصلة عشرية عائمة (٨ بايت)	من **D308**١,٧٩٧٦٩٣٧٣٤٨٦٢٣٠٨ – الى **D**٤,٩٤٠٦٥٦٤٥٨٤١٢٤ ٣٢٤ (قيم سالبة) من **4.94065645841247D324** إلي **D308** ١,٧٩٧٦٩٣١٣٤٨٦٢٢٣٢(قيم موجبة)
Currency	عدد ذو فاصلة عشرية ثابتة **Fixed Point**	من – ٩٢٢٣٣٧٢٠٣٦٨,٥٤٧٧٥٨٠٨ إلي ٩٢٢٣٣٧٢٠٣٦٨٥٤٧٧,٥٨٠٧
String	مجموعة من الحروف	من ٠ إلي ٦٥,٥٠٠ حرف تقريباً
Variant	الوقت / التاريخ، عدد ذو علامة عشرية عائمة، أو سلسة حروف	التاريخ: من ١ يناير١٠٠ إلى ٣١ ديسمبر ٩٩٩٩ و الأعداد مثل **Double** و الحروف مثل **String**

الإعلان عن المتغيرات **Variables Declaration**:

يجب تعريف فيجوال بيسك بأسم المتغير ونوعه، مثل:

Dim StudentName As String

أمر **Dim** يخبر فيجوال بيسك أننا نريد الإعلان عن متغير، وكلمة **StudentName** هي اسم المتغير وكلمة **String** هي نوع المتغير. أي أننا نخبره أننا نريد استخدام المتغير **StudentName** كمتغير حرفي. فيقوم فيجوال بيسك بحجز الذاكرة الضرورية لهذا المتغير.

والإعلان عن المتغيرات ليس إجباري. فإذا استخدمت متغيراً دون الإعلان عنه، فان فيجوال بيسك يحجز له مكان بالذاكرة ويقوم بتهيئته تلقائياً علي أنه متنوع **Variant**.

وقد يؤدى عدم الإعلان عن المتغيرات أحيانا إلى أخطاء في برنامجك. فمثلاً في السطرين التاليين:

UserName$ = InputBox$(ادخل اسمك من فضلك)
Print UseName$

يقوم أمر **InputBox$** يطلب اسم المستخدم ويضعه في المتغير **UserName$**. والسطر الثاني يعرض هذا الاسم علي النافذة. إلا أنك لو جربت هذين السطرين، ستجد أن السطر الثاني لن يطبع شيئاً. لوجود خطأ في اسم المتغير في السطر الثاني: فهو هنا **UseName$** وليس **UserName$** كما في السطر الأول وبالتالي سيعتبره فيجوال بيسك متغيراً جديداً وليس نفس المتغير الموجود في السطر الأول. رغم ذلك لن يعترض فيجوال بيسك وسيقوم بطباعة قيمته علي النافذة، ولأنه متغير جديد فهو لا يحتوى علي أي شئ وبالتالي لن يطبع شئ علي النافذة. هذه الأخطاء تحدث كثيراً وهى لا تؤدى إلي ظهور رسالة خطأ ولا يتوقف عندها البرنامج وبالتالي يصعب اكتشافها.

ويمكنك إخبار فيجوال بيسك بعرض رسالة خطأ عند ورود أي متغيرات لم يسبق الإعلان عنها وذلك بالأمر التالي:

Option Explicit.

وتضعه في قسم الإعلان **Declaration Section** في النافذة أو في الكود الذي يراد استخدامه فيه. بعد استخدام هذا الأمر، إذا قابل فيجوال بيسك متغيراً جديداً لم يسبق الإعلان عنه فإنه يتوقف ويعرض رسالة خطأ وفيها اسم المتغير. إن كنت متأكدا أنك أعلنت عن هذا المتغير من قبل، فلابد أن هناك خطأ هجائي في اسمه، يمكن تصحيحه ومن ثم استكمال البرنامج.

أمر **Option Explicit** يعمل علي الملف أو النافذة التي ورد فيها فقط، لذلك يجب وضعه في قسم الإعلان في كل نافذة أو ملف تريد مد تأثيره عليه. وإذا رغبت في استخدامه بكل المشروع، يمكنك تغيير الخيارات الخاصة بالمشروع من قائمة **Option** بأمر **Environment** ثم تغير الخيار **Require Variable Declarations**

إلى Yes ولن تحتاج بعد ذلك إلى اصدار أمر **Option Explicit** وانما سيقوم فيجوال بيسك بوضعه تلقائياً في كل ملف أو نافذة جديدة. ويتعين عليك بعد هذا أن تعلن عن كل المتغيرات التي ستقوم باستخدامها.

ويتم الإعلان عن المتغيرات باستخدام أحد الأوامر التالية:

Dim, Global, Static

ويتم اختيار الأمر المناسب من هذه الأوامر حسب مدى المتغير وطبيعته.

الالتزام بنوع المتغير:

سواء أعلنت عن المتغير أم لا، يجب عليك الالتزام بنوعه عند تغيير قيمته. فإذا أعلنت عن المتغير **CompanyName** كمتغير حرفي ثم حاولت أن تضع فيه عدد أو أن تجري عليه عملية حسابية مثل:

CompanyName = 60

سيعرض لك فيجوال بيسك رسالة خطأ لعدم الالتزام بنوع المتغير.

ومعنى هذه الرسالة أن هناك عدم توافق بين القيمة التي تريد إدخالها وبين نوع المتغير وعليك تغيير القيمة إلي قيمة حرفية أو تعدل نوع المتغير إلى متغير عددي.

والنوع الأخير من أنواع المتغيرات **Variant** يمكنه تخزين قيماً مختلفة ويستخدم كأي نوع من أنواع المتغيرات الاخري، ويمكنه كذلك التحويل تلقائياً بين أنواع المتغيرات بدون إصدار رسالة الخطأ السابقة. إذا لم تحدد النوع فإنه يصبح **Variant**.

فمثلاً:

Dim MyValue, Variant
My Value = "162"

هنا يصبح المتغير متغيراً حرفياً.

MyValue = MyValue * 2

يتحول المتغير إلى عددي وتجري عليه العملية الحسابية.

ويجري الأمر الأخير عملية حسابية علي متغير حرفي، ولأن هذا المتغير من نوع Variant، سيقوم فيجوال بيسك بتحويله تلقائياً إلى متغير عددي ويضع فيه القيمة ١٦٢ ثم يضربها في ٢ ويضعها في نفس المتغير. كذلك:

Variant MyValue = 153

متغير عددي من نوع رقمي.

يتحول المتغير إلى حرفي

MyValue = MyValue + "M"

هنا يحول فيجوال بيسك هذا المتغير إلى متغير حرفي ثم يضع فيه القيمة **153M**.

مدى المتغيرات وعمرها:

يحدد مدى المتغيرات **Scope Of Variables** الأماكن التي يمكن أن تستخدم فيه المتغير. ويحدد عمر المتغيرات،**Lifetime of Variables** مدة بقائها في الذاكرة.

وتنقسم المتغيرات من حيث مداها وعمرها إلى ثلاثة أقسام:

١- المتغيرات العامة **Global Variable**.

٢- المتغيرات علي مستوى الملف أو النافذة **Module Level Variable**

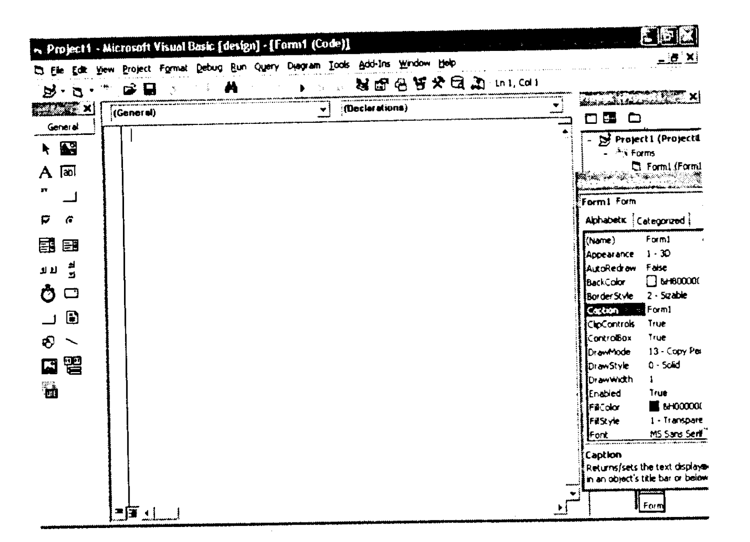
٣- المتغيرات المحلية علي مستوى الإجراء **Procedure Level Variable**.

١- المتغيرات العامة:

المتغير العام **Global Variable** هو المتغير الذي يمكنك استخدامه من اى مكان في البرنامج، فمداه يشمل جميع البرنامج بجميع ملفاته ونوافذه. ويبقي في الذاكرة طوال فترة عمل البرنامج ولا يحذف من الذاكرة إلا بعد انتهاء البرنامج.

يجب أن تضع الإعلان عن المتغير العام في ملف برمجة **Code Module** وليس في نافذة. وملف البرمجة هو ملف لا يحتوى علي نوافذ، ويستخدم فقط لوضع المتغيرات والإجراءات العامة وينتهي بالامتداد **bas.***.

لعمل ملف برمجة جديد أنقر اختار أمر **New Project** من قائمة **File**. عندها سيضيف فيجوال بيسك ملفاً جديداً للمشروع ويسميه**project1.bas** وسيعرض لك نافذة البرمجة الخاصة كما يلي:



إنشاء ملف برمجة:

يمكنك إدخال المتغيرات التي ترغب في جعلها متغيرات عامة. ويجب أن تسبقها كلمة **Global** مثل:

Global UserName As String

بعد هذا الإعلان يمكن استخدام المتغير**UserName** من أي مكان داخل البرنامج وستجد أنه يحتفظ بقيمته طوال فترة عمل البرنامج.

بعد الانتهاء من الكتابة داخل ملف البرمجة، أغلقه بالنقر علي صندوق التحكم الموجود في صف العنوان الخاص به نقراً مزدوجاً. أو تأكد من أنه النافذة النشطة ثم اضغط علي مفتاحي **Alt + F4**. وبعد غلق ملف البرمجة والرغبة في الكتابة فيه مرة أخري، انتقل إلى نافذة المشروع **Project Window** ثم انقر اسمه نقراً مزدوجاً. فيقوم فيجوال بيسك بفتحه وعرض محتوياته إمامك.

٢- المتغيرات علي مستوى الملف أو النافذة:

Module-Level Variables:

المتغيرات علي مستوى الملف أو النافذة هي متغيرات محدودة بالملف أو النافذة التي أعلنت عنها فيها. فيمكنك استخدامها من اى إجراء داخل الملف أو النافذة، ولكن لا يمكن استخدامها من اى مكان خارجها. وهذا النوع يبقي طوال فترة عمل البرنامج ولا يحذف من الذاكرة إلا بعد انتهاء البرنامج. والفرق بينها وبين المتغيرات العامة هو في مداها فقط، فبينما يمتد مدى المتغيرات العامة ليشمل البرنامج كله، تكون هذه مقصورة علي الملف أو النافذة التي أعلنت داخلها ولا يمكن استخدامها خارجها.

ويمكنك استخدام متغير علي مستوى النافذة عندما يكون المشروع مكون من عدة نوافذ وترغب في أن تكون كل نافذة مستقلة بزاتها. في هذه الحالة أعلن عن المتغيرات الخاصة بكل نافذة داخلها بحيث لا تعتمد النافذة في عملها علي متغيرات خارجية. مما يمكنك من إعادة استخدام النافذة في مشروع آخر دون الحاجة إلى إجراء تعديلات عليها.

لإدخال المتغيرات علي مستوى ملف البرمجة يتم استخدام أمر **Dim** مثل:

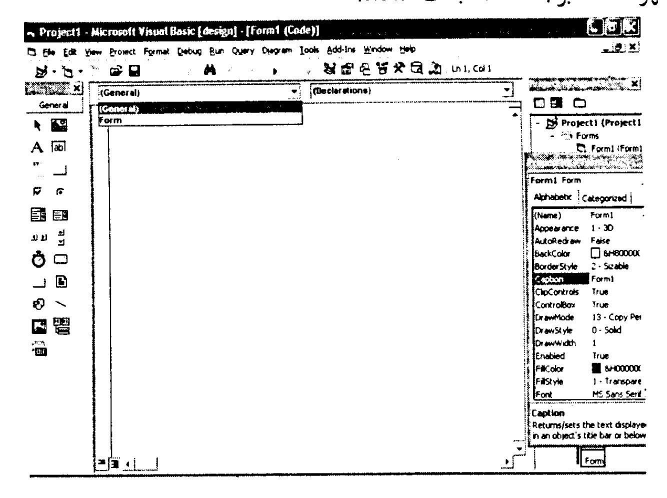
Dim RecordSize As Long

فتفتح ملف البرمجة، وتبدأ في إدخال اسم المتغير مباشرة. والفرق بين المتغير العام **Global** والمتغير علي مستوى الملف في هذه الحالة هو في الأمر المستخدم للإعلان عنه. ففي المتغير العام نستخدم كلمة **Global** وفي المتغير علي مستوى الملف نستخدم كلمة **Dim**.

لإدخال المتغيرات علي مستوى النافذة:

- انقر فوق أي مكان علي النافذة المطلوب إضافة المتغير لها.

ستظهر نافذة البرمجة الخاصة بتلك النافذة.



- انقر زرار السهم الموجود في يمين خانة الأداة (الخانة أليسري في نافذة البرمجة). ستنسدل القائمة وبها الأدوات الموجودة علي النافذة.
- اختار كلمة **General** من القائمة.

إذا كان لديك أدوات كثيرة علي النافذة، فقد لا تظهر كلمة **general** عند النقر علي الزرار. في هذه الحالة، استخدم عمود التحرك الرأسي لتنتقل إلى أعلي القائمة. ستجد

كلمة general وأن الاسم الموجود في خانة الأحداث قد تغير إلى كلمة declarations.

فيمكنك إدخال المتغيرات علي مستوى النافذة. ويتم الإعلان عنها باستخدام أمر Dim.

مثل:

Dim FileName as String

٣- المتغيرات المحلية علي مستوى الإجراء Procedure Level Variable:

المتغيرات المحلية علي مستوى الإجراء **Procedure** هي أقل المتغيرات في مداها، فهى مقصورة علي الاجراء الذي أعلن عنها فيه فقط، ولا يمكنك استخدامها في أي مكان غيره. ويتم الإعلان عن هذا النوع من المتغيرات داخل الإجراء نفسه. وبالرغم من امكان الإعلان عن المتغير في اى مكان داخل الإجراء قبل النقطة التي ستستخدمه فيها، إلا أنه من المستحسن الإعلان عن جميع المتغيرات التي ستستخدمها داخل الإجراء دفعة واحدة في بدايته. وبهذا تستطيع البحث عن المتغيرات بسهولة عند رغبتك في تعديل أي منها، بدلاً من البحث عنها في كل الإجراءات.

ويختلف عمر المتغيرات حسب الأمر الذي ستستخدمه في الإعلان عنها. فإذا أعلنت عنها باستخدام أمر **Dim** فإن فيجوال بيسك يزيلها من الذاكرة بعد انتهاء الإجراء وبالتالي فإنها تفقد قيمتها، ويتم استرجاع الذاكرة التي كانت تحتلها. وإذا استدعيت الإجراء مرة أخري، فإن فيجوال بيسك يعيد تهيئتها وحجز الذاكرة لها من جديد. وهذا النوع من المتغيرات مفيد للمتغيرات المؤقتة التي لن تستخدمها خارج الإجراء وهو أقلها استهلاكاً للذاكرة.

وقد ترغب في الاحتفاظ بقيمة متغير بعد الانتهاء من الإجراء في هذه الحالة تستخدم أمر **Static** في الإعلان عنها. مثل:

Sub Command1_Click ( )

```
Static Count As Integer
Count = Count + 1
Label1.Caption = Str$ (Count)
```

بهذا الشكل، يظل المتغير **Count** موجوداً في الذاكرة ومحتفظا بقيمته حتى بعد أن ينتهي الإجراء. والفرق بينه وبين المتغير العام أو المتغير علي مستوى النافذة في مداه فقط. ففي كل الأحوال يظل مداه مقصوراً علي الإجراء الذي أعلن عنه فيه.

في المثال السابق، إذا نقرت زرار الأوامر **Command1** يقوم فيجوال بيسك بتنفيذ الإجراء ويضيف ١ إلي المتغير **Count** ويضع تلك القيمة في عنوان **Label1**. ولأننا أعلنا عن المتغير باستخدام أمر **Static** فإنه يظل محتفظاً بقيمته بعد انتهاء الإجراء. وبهذا تظل قيمته تزيد في كل مرة تنقر فيها فوق زرار الأوامر.

ولو أعلنا عنه باستخدام أمر **Dim**، ستظل قيمته عند ١ ولا تزيد أبداً لأنه في كل مرة ينتهي الإجراء يقوم فيجوال بيسك بإزالة المتغير من الذاكرة وبالتالي يفقد قيمته. وعند بدء الإجراء من جديد، يقوم بتهيئته إلى صفر وعندما ينفذ أمر:

Count = Count + 1

تصبح تلك القيمة ١ فقط.

وإذا أردت جعل كل المتغيرات في إجراء معين تحتفظ بقيمتها ولا تحذف من الذاكرة، استخدم أمر **Static** في أول الإجراء، مثل:

Static Sub MyProc ( )

في هذه الحالة تبقي أي متغيرات تعلن عنها في داخل هذه الإجراء في الذاكرة طوال فترة عمل البرنامج ولن تحتاج إلى كتابة أمر **Static** عند الإعلان عن كل متغير.

ملحوظة:

من **الأفضل** تحديد مدى المتغيرات التي تريد استخدامها وعمرها، منعاً للخلط بينها ولمنع استهلاك الذاكرة بدون داع. فإذا احتجت إلي استخدام متغير ما في أكثر من نافذة في

داخل المشروع، أعلن عنها كمتغير عام، وإذا كنت تحتاج إلى استخدامه داخل نافذة واحدة فقط، فمن الأفضل أن تعلن عنه داخل هذه النافذة. أما إذا كنت تحتاج إليه مؤقتاً أثناء الإجراء فقط، فأعلن عنه داخل هذا الإجراء.

الثوابت **Constants**:

علي عكس المتغيرات، فإن الثوابت قيم ثابتة لا تتغير أثناء تنفيذ البرنامج، والهدف منها هو إعطاء أسماء لها معنى للأرقام التي تستخدمها في برنامجك مما يساعدك علي تذكر الهدف من استخدامها ويسهل عليك تغييرها.

لنفترض أن لديك عشرين موظفاً تريد إدخال بياناتهم. يمكنك كتابة عدد الموظفين مباشرة في كل مرة تحتاج إلى ذلك. مثل:

```
For Count = 1 to 20
.........
........
Next
```

لنفترض أنك كتبت أمراً مثل هذا في ستة أو سبعة مواضع داخل البرنامج، ثم أردت بعد ذلك أن تغير عدد الموظفين من ٢٠ إلي ٥٠. في هذه الحالة ستضطر إلى البحث عن كل المواضع التي ذكرت فيها رقم ٢٠ وتغيره إلى ٥٠، وإذا نسيت موضعاً سيؤدى ذلك إلى خلل في عمل البرنامج.

لذلك، استخدم ثابتاً **Constant** وضعه في بداية البرنامج أو الإجراء مرة واحدة وتضع فيه العدد الذي ترغب فيه، مثل:

```
Const NUM_OF_EMPLOYEES = 20
```

فيتحقق أمران: الأول أن الثابت

NUM_OF_EMPLOYEES

اسهل في تذكر معناه عن الرقم ٢٠.

والثاني: أنك لو اردت تغيير عدد الموظفين من ٢٠ إلى ٥٠ مثلاً، فكل ما عليك هو أن تغير قيمة الثابت في المكان الذي أعلنت عنه فقط، مثل:

```
Const NUM_OF_EMPLOYEES = 50
```

بدلاً من البحث عن الرقم ٢٠ في كل المواضع التي ذكر فيها وتغييره.

وهناك استخدم آخر للثوابت، تحتاج أحياناً إلى استخدام بعض الأرقام المحددة سلفاً في برنامجك وقد تكون هذه الأرقام صعبة التذكر وقد تخلط بينها. فمثلاً يستخدم فيجوال بيسك نظاماً خاصاً لتحديد رموز درجات الألوان. فإذا أردت أن تغير لون أداة إلى الأزرق أثناء تشغيل البرنامج، تكتب:

```
Form1. BackColor = &HFF0000
```

يستخدم فيجوال بيسك الرمز &HFF0000 للأزرق، والرمز &HFFFFFF للأبيض والرمز &HFF& للأحمر، سيكون من الأسهل إعطاء هذه الأرقام رموزاً ثابتة مثل:

```
Const  BLACK = &H0&
Const  RED = &HFF&
Const  GREEN = &HFF00&
Const WHEITE = &HFFFFFFF
Const Blue = &HFF0000
```

ويتم استخدام أسماء هذه الثوابت الواضحة المعنى بدلاً من الرموز المركبة لتغيير لون أداة ما. فمثلاً:

```
Form1.BackColor = BLUE
```

ويمكن أن تكون الثوابت ثوابت حرفية أيضاً. مثل:

```
Const PASSWORD = FAY
```

في هذه الحالة يمكنك استخدام الثابت PASSWORD في اى مكان تريد استخدام العبارة FAY فيه. مثل:

Print PASSWORD

اختيار اسم الثابت:

يخضع اختيار اسم الثابت لنفس الشروط التي ذكرناها عند اختيار اسم المتغير.

مدى الثوابت:

يتحدد مدى الثوابت بالمكان الذي أعلنت عنها فيه. وهى تتبع نفس القواعد التي تحدد مدى المتغيرات. فإذا أردت استخدام الثابت من أي مكان في البرنامج. فيجب أن تعلن عنه في ملف برمجة CodeModule وتسبق إعلانه بكلمة Global مثل:

Global Const USER_NAME = عمرو

أما إذا أردت استخدام الثابت في نافذة واحدة أو ملف واحد دون سائر البرنامج، فتعلن عنه في قسم الإعلان داخل هذه النافذة أو الملف بدون كلمة Global مثل:

Const NUM_OF_LINES = 78

وإذا أردت استخدامه بشكل مؤقت داخل إجراء واحد. فتعلن عنه داخل هذا الإجراء بنفس الطريقة السابقة.

الإجراءات (الموديول) Procedures:

الإجراء هو جزء مستقل من البرنامج له بداية ونهاية ويحتوى علي مجموعة من الأوامر والعبارات والمتغيرات والثوابت. ويمكنك استدعاؤه بمجرد ذكر اسمه في برنامجك.

فيتقل البرنامج لهذا الجزء وينفذ الأوامر الموجودة فيه وبعد أن ينتهي منها يعود إلى السطر التالي لأمر الاستدعاء.

أهمية الإجراءات:

مع كبر حجم البرنامج الذي تكتبه، ستجد أجزاء من البرنامج تحتاج إلى تنفيذها أكثر من مرة في أماكن مختلفة. فمثلاً قد تحتاج إلى سؤال المستخدم عن اسم ملف لحفظ البيانات ثم تتأكد إن كان الاسم الذي ادخله المستخدم سليماً أم لا، فإن لم يكن سليماً تعاود الكرة وتسأله عن اسم صحيح، وهكذا حتى يدخل اسماً صحيحاً للملف أو يلغي الأمر. لنفرض انك تحتاج إلى هذا الجزء في أكثر من مكان داخل برنامجك. يمكنك كتابة هذا الجزء مرة واحدة ثم نسخه في كل مرة تحتاج إليه وتلصقه في المكان الجديد. مما يؤدى إلى كبر حجم البرنامج بدون داع. وإذا رغبت في تعديل هذا الجزء وتحسينه، ستضطر إلى البحث عنه في كل مكان وضعته فيه وإجراء التحسينات عليه، وهى مسالة شاقة. وقد تنسي تعديله في مكان ما مما يؤدى إلى خلل في البرنامج.

لذلك تكتب مجموعة الأوامر المتكررة في إجراء عام **General Procedure** ثم تستدعى هذا الأجراء كلما احتجت له. وبهذا نتخلص من مشكلة كبر حجم البرنامج لأن مجموعة الأوامر لن تتكرر في اكثر من مكان، كما أن صيانة هذا الجزء وتعديله ستصبح سهلة لأن الأوامر موجودة كلها في مكان واحد.

وحتى إذا لم تحتاج إلى هذا الجزء في أكثر من موضع داخل البرنامج، فإن تقسيم البرنامج إلى إجراءات مستقلة يقوم كل منها بأداء جزء من البرنامج أو بحل مشكلة معينة، هو التقليد المفضل لدى معظم المبرمجين. لأنه يؤدى إلى تنظيم سير البرنامج بصورة أفضل ويجعل صيانته أسهل.

كيفية كتابة الإجراء:

إذا رغبت في كتابة إجراء. أبدأ أولاً بتحديد اسم الإجراء. لاحظ ضرورة إتباع نفس قواعد اختيار أسماء المتغيرات والثوابت. لنفرض مثلاً أنك تريد أن تسمى الإجراء GetFileName، ابدأ بوضع إطار للإجراء مثل:

```
Sub GetFileName ( )

End Sub
```

كلمة Sub تعلن عن بداية الإجراء وكلمة End Sub تعلن نهايته. بعد ذلك أكتب مجموعة الأوامر التي ترغب في إضافتها للإجراء بين هذين السطرين. فمثلاً:

```
Sub GetFileName ( )
Dim FileName as String
FileName = أدخل اسم الملف من فضلك
If Dir (FileName) then
 InputBox$

Response = MsgBox
.......
End if
End Sub
```

كيفية استدعاء الإجراء:

بعد الانتهاء من كتابة الإجراء، يمكنك استدعاؤه بذكر اسمه فقط. فمثلاً:

```
Sub CmdSave_Click( ) لحفظ الملف
GetFileName, مجرد ذكر اسم الإجراء يستدعيه
........
End Sub
```

في هذا المثال، يتم استدعاء الإجراء كلما نقر المستخدم زرار الأوامر المسمى CmdSave.

تمرير المتغيرات للإجراءات:

ستحتاج إلى تمرير متغير أو أكثر إلى الأجزاء المختلفة بالبرنامج. فإذا رغبت في كتابة إجراءاً لفتح ملف معين وقراءة محتوياته ووضعها في خانة نص مثلاً. فيحتاج الإجراء إلى معلومتين: الأولي هي اسم الملف والثانية هي اسم خانة النص. لذلك عدل راس الإجراء بحيث تخبر فيجول بيسك عن المتغيرات التي يريدها هذا الإجراء. فمثلاً:

Sub ReadFile (FileName As String, TextBox As Control).

هنا نخبر فيجوال بيسك أن الإجراء **ReadFile** يتوقع أن نمرر له متغيرين. الأول متغير حرفي **String** والثاني هو أداة من أدوات التحكم **Control**.

بعد ذلك عندما ترغب في استدعاء الإجراء، يجب أن تمرر له متغيرين من نفس النوعين المعلن عنهما في راس الإجراء. مثل:

ReadFile "C:\report.txt , Text1"

هنا نستدعى الإجراء **ReadFile** ونمرر له اسم الملف واسم خانة النص. سيضع فيجوال بيسك اسم الملف في المتغير **FileName** المذكور في راس الإجراء، ويضع اسم خانة النص في المتغير **TextBox**. ويمكن للإجراء نفسه **ReadFile** أن يقرأ المعلومات التي مررناها له عن طريق استخدام المتغيرين **TextBox, FileName**.

وعند تمرير المتغيرات إلى الإجراء يجب ملاحظة:

- أن يكون عدد المتغيرات عند استدعائك للإجراء هو نفس عدد المتغيرات الذي حددته في راس الإجراء.
- أن تكون نوعية المتغيرات عند الاستدعاء هي نفسها في راس الإجراء.

وعند مخالفة اى من هذين الأمرين سيصدر لك فيجوال بيسك رسالة خطأ أثناء تنفيذ البرنامج.

الإجراءات التي تحسب قيمة وتعود بها إلى البرنامج:

يمكن للإجراء أن يعود بقيمة إلى الجزء الذي استدعاه. فمثلاً يمكن كتابة إجراء ينفذ عدة عمليات حسابية ثم يعود إليك بنتيجتها. فيتم الإعلان عن الإجراء باستخدام كلمة دالة **Function** بدلاً من كلمة **Sub**، مثل:

```
Function CalculateNum ( )

End Function
```

CalculateNum هو اسم الإجراء. ولأننا استخدمنا كلمة **Function**، فان فيجوال بيسك يعرف أن هذا الإجراء سيعود بقيمة معينة. وإذا احتجت أن تمرر لهذا الإجراء متغيرات معينة، فضعها بين القوسين، مثل:

```
Function CalculateNum (A As Integer, B As Integer)
```

بعد ذلك تكتب الأوامر والعبارات التي تريد من الإجراء أن يقوم بها. وفي نهاية الإجراء ضع القيمة التي تريد أن يرجع بها الإجراء في اسم الإجراء نفسه، مثل:

```
Function CalculateNum (A As Integer, B As Integer)
القيمة التي سيرجع بها الإجراء
CalculateNum = A*B/2+52
End Function
```

بعد أن يقوم الإجراء بالعمليات المطلوبة، يضع ناتج هذه العمليات في اسم الإجراء نفسه حتى يعود بها إلى الجزء الذي استدعاه من البرنامج.

لاستدعاء الإجراء اكتب أمراً مثل:

```
MyNum = CalcuateNum (5,7)
```

تستدعى الإجراء **CalculateNum** ونمرر له الرقمين المذكورين حيث يدخل ٥ في **A** و ٧ في **B** ونستقبل القيمة التي سيعود بها في المتغير **MyNum**.

ويجب استخدام متغير يتناسب مع نوع القيمة التي يعود بها الإجراء وإلا عرض فيجوال بيسك رسالة خطأ. فإذا كان الإجراء يعود بمتغير عددي، يجب أن تستخدم متغيراً عددياً عند استدعائه. وإذا كان يعود بمتغير حرفي، يجب أن تستخدم متغيراً حرفياً.

كيفية عمل الإجراءات:

عندما تستدعى إجراء ما، يتفرع البرنامج إلى هذا الإجراء إلى أن ينفذ جميع التعليمات الموجودة فيه ثم يعود إلى الأمر التالي لأمر استدعائه. فمثلاً:

1) A$ = "Visual Basic"	Sub UpdateList (Item As String) List1.AddItem.Item
2) UpdateList A$	....
3) Print تم عمل اللازم	End Sub

ينفذ البرنامج العبارة في السطر الأول ثم يستدعى في السطر الثاني الإجراء **UpdateList** ويمرر له المتغير **A$** ويبقي هناك حتى ينتهي من الإجراء ثم يعود لينفذ الأمر في السطر الثالث.

وتنتهي الإجراءات والوظائف بتنفيذ آخر عبارة أو أمر فيها وتصل إلى عبارة **End Sub** أو عبارة **End Function**، ويمكنك الخروج من الإجراءات قبل أن تصل إلى نهايتها باستخدام أمر **Exit Sub** بالنسبة للإجراءات التي لا تحسب قيمة أو الأمر **Exit Function** بالنسبة للإجراءات التي تحسب قيمة. وهذان الأمران يوفران لك نهاية بديلة للإجراءات ولهذا فهما يردان في الغالب داخل تركيب شرطي يختبر قيمة معينة ثم تتخذ قرار الخروج من الإجراء أو البقاء فيه بناء علي النتيجة مثل:

```
Sub ReadFile (FileNo As Integer)
For Counter = 1 To 100
Input # FileNo, A$
If A$ = "Done" Then
MsgBox (تم قراءة كل البيانات المطلوبة)
```

```
Exit sub
End if
Next
.......
End Sub
```

في هذا المثال، نقوم بقراءة محتويات ملف يحتوى علي ١٠٠ سجل أو أقل، وآخر سجل فيه هو عبارة **Done**. نستخدم التكرار **For.........Next** ليكرر القراءة مائة مرة. ولأن الملف يمكن أن يحتوى علي عدد مدخلات اقل من ١٠٠، فإننا نختبر القيمة التي قرأها، فان كانت هي العبارة **Done**، فمعنى هذا أننا وصلنا إلى نهاية الملف، وبالتالي لا داعي لإكمال التكرار. لذلك نستخدم الأمر **Exit Sub** للخروج من الإجراء قبل الوصول إلى نهايته العادية.

مدى الإجراءات:

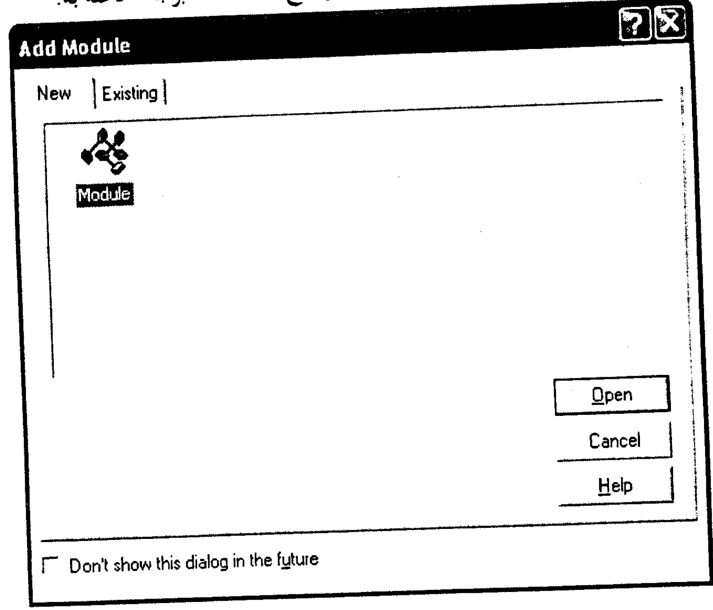
للإجراءات مدى مثل المتغيرات والثوابت. ويتحدد مدى الإجراءات حسب المكان الذي تضعها فيه. فإذا وضعتها في قسم **General** داخل النافذة، يصبح مداها مقصوراً علي تلك النافذة ولا تستطيع أن تستدعيها من خارجها. أما إذا وضعتها في ملف برمجة **Code Module**، فإنها تصبح عامة **Global** ويمكن استدعاءها من أي مكان في البرنامج.

مكان وضع الإجراءات(الموديول):

لإدخال الإجراءات؛ حدد أولاً النافذة الذي تريد وضعها فيه، ثم احضر نافذة البرمجة الخاصة بهذه النافذة أو الملف.

إذا كنت ستضع الإجراءات داخل نافذة، انقر علي اى مكان فوق هذه النافذة لتستدعى نافذة البرمجة الخاصة بها.

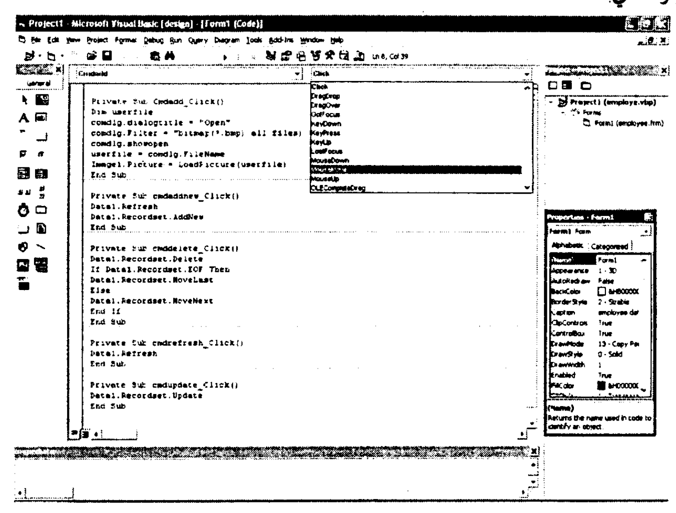
أما إذا كنت ستضعها في ملف برمجة جديد، انقر الرمز من صف الأدوات أو انقر الأمر New Module من قائمة مشروع Project فيضع فيجوال بيسك ملف برمجة جديد في المشروع ويسميه Module.bas ويفتح لك نافذة البرمجة الخاصة به.



في هذا المثال، نقوم بقراءة محتويات ملف يحتوى علي ١٠٠ سجل أو أقل، وآخر سجل فيه يحتوى على Done. نستخدم التكرار For... Next ليكرر القراءة مائة مرة. ولان الملف يمكن أن يحتوى علي عدد مدخلات اقل من ١٠٠، فإننا نختبر القيمة التي قراها، فإن كانت العبارة Done، فمعنى هذا أننا وصلنا إلى نهاية الملف، وبالتالي لا داعي لإكمال التكرار. لذلك نستخدم الأمر Exit Sub للخروج من الإجراء قبل الوصول إلى نهايته العادية.

وإذا كان ملف البرمجة الذي ستضع فيه الإجراء موجوداً في المشروع، انقر عليه في نافذة المشروع نقراً مزدوجاً لتظهر نافذة البرمجة الخاصة به. تأكد أن نافذة البرمجة هي النافذة

النشطة ثم اختار أمر **Code** من قائمة **View** سيعرض لك فيجوال بيسك صندوق الحوار التالي:



حدد نوع الإجراء المطلوب. فإذا أردت إجراء يحسب قيمة، اختار كلمة **Function** وإذا أردت إجراء لا يحسب قيمة، اختار كلمة **Sub**. ثم اكتب اسم الإجراء الجديد في خانة **Name** وانقر زرار **OK**.

سيقوم فيجوال بيسك بمسح محتويات نافذة البرمجة ويضع لك إطاراً فارغاً للإجراء أكتب فيه الأوامر.

هناك طريقة أخري أسرع لإدخال الإجراءات: اكتب راس الإجراء الذي ترغب في إضافته في اى مكانٍ في نافذة البرمجة. سوف يقوم فيجوال بيسك بمسح محتويات النافذة ووضع إطار للإجراء الجديد.

تعديل الإجراءات:

بعد الانتهاء من كتابة إجراء معين، يمكن أن تعود إليه للتعديل فيه، نفذ ما يلي:

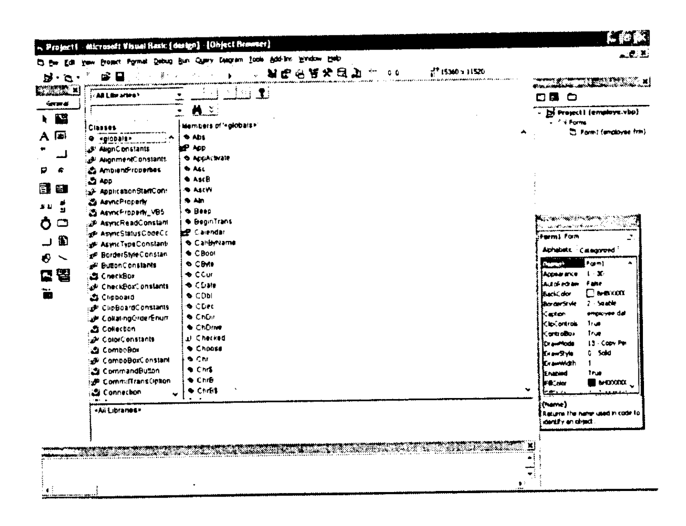
١– احضر نافذة البرمجة التي أدخلت فيها الإجراء.

٢- اختار **General** من خانة الأداة (الخانة اليسري في اعلي النافذة).

٣- انقر زرار السهم الموجود في يمين خانة الأحداث (الخانة اليمنى في اعلي النافذة). ستنسدل قائمة بكل الإجراءات الموجودة في النافذة أو الملف.

٤- استخدم عمود التمرير في البحث عن الإجراء الذي تريده، ثم انقر عليه. سيحضر فيجوال بيسك هذا الإجراء إلى نافذة البرمجة حيث يمكنك إجراء التعديلات التي ترغب فيها.

هناك طريقة أخري لاستعراض جميع الإجراءات الموجودة في البرنامج والتنقل بينها وتحريرها. اضغط مفتاح **F2** أو اختار أمر **Object Browser** من قائمة **View**. سيعرض عليك فيجوال بيسك الصندوق التالي:



هذا الصندوق ينقسم إلى قسمين: الأول **Modules** يحتوى علي أسماء جميع النوافذ وملفات البرمجة الموجودة في المشروع.

- اختار اسم الملف الذي تريد استعراض الإجراءات الموجودة فيه.

ستجد أن القائمة الثانية في الصندوق تحتوى علي الإجراءات الموجودة في الملف الذي اخترته. يمكنك استعراض محتويات القائمة لتتعرف علي الإجراءات الموجودة فيها.

- عندما تصل إلى إجراء تريد قراءة محتوياته أو تعديلها، انقر فوق اسمه ثم انقر **OK**.

سيفتح فيجوال بيسك نافذة البرمجة الخاصة بهذا الإجراء ويعرضه لك.

التعامل مع المصفوفات Matrix / Arrays في الفيجوال بيسك

تعريف المصفوفة

- عبارة عن متغيرات مفهرسة Indexed Variables تحتوي على بيانات عديدة من نفس النوع

Data Type

ولكل مصفوفة لها اسم واحد يمكن استخدامه للرجوع إلى أي عنصر فيها وذلك باقتران هذا الاسم بدليل يمثل مكان العنصر فيها ، ويمكن انشاء مصفوفة لإحتواء أي نوع من أنواع البيانات مثل النصوص والأعداد الحقيقية والصحيحة وغيرها ، فأنواع البيانات المتوفرة في الفيجوال بيسك هي

Data Type in VB {Byte, Boolean, Integer, Long, Single, Double, Currency, Decimal, Date, Object, String, Variant, User-defined }.

ويساعد استخدام المصفوفات في البرمجة في اعداد أكواد قصيرة وبسيطة ذات قوة كبيرة لأنه يمكن بناء دورات Loops تتعامل بكفاءة مع المصفوفات مهما كان عدد عناصرها وذلك باستخدام دليل العنصرIndex Number .

الخصائص الأساسية للمصفوفة في الفيجوال بيسك

١. اسم المصفوفة يمثل عنوان Address في الذاكرة ؛ ولا يمكن تغييره أثناء تنفيذ البرنامج.

٢ يمكن الإعلان عن مصفوفة لأي نوع من أنواع البيانات بما في ذلـــك الأنـــواع المعرفة من قبل المســـتخدم **User-defined type** والـــ **Object Variables**.

٣. كل وحدة بيانات منفردة في المصفوفة تسمى عنصر **Element**

٤. جميع العناصر تكون من نفس النوع إلا في حالة الإعلان عن المصفوفة كمصفوفة بيانات متنوعة **Variant Data Type**.

٥. جميع العناصر تكون مخزنة على التتابع في ذاكرة الحاسب ودليل أول عنصر هو الصفر كقيمة مفترضة **Default** ، ويمكـــن جعلـــه ١ باســـتخدام جملـــة **Option Base**.

٦. يحدد موقع العنصر بواسطة رقم الصف ورقم العمود الذي يقع فيه.

٧. لكل مصفوفة حـــداً أعلـــى **Upper bound** ، وحـــداً أدنى **Lower bound**؛ وعناصر المصفوفة تكون محصورة بين هذين الحدين.

٨. من الممكن أن تكون المصفوفة ذات بعد واحد أو متعددة الأبعاد.

٩. هناك نوعين من المصفوفات

- *المصفوفة ذات الحجم الثابت* ***Fixed-size array*** وهي التي يظل حجمها ثابت أثناء تشغيل البرنامج.
- *المصفوفة ذات الحجم المتغير* ***Dynamic-size array*** وهي التي يمكن تغيير حجمها أثناء تشغيل البرنامج.

سندرس ما يلى فى موضوع المصفوفات :

- كيفية تعريف مصفوفة ذات حجم ثابت **Fixed-size Array**
- كيفية تحديد الحدين الأعلى والأدنى لأي مصفوفة.
- المصفوفات متعددة الأبعاد **Multi-Dimensional Arrays**
- جملة تغيير رقم بدء دليل المصفوفة **Option Base**
- استخدام الدورات **Loops** لمعالجة المصفوفات.
- كيفية تعريف مصفوفة ذات حجم متغير **Dynamic Array**

- حفظ محتويات الـ Dynamic Arrays عند إعادة تعريفها.
- جملة ReDim .
- كيفية تعريف مصفوفة ذات حجم متغير Dynamic Array
- حفظ محتويات الـ Dynamic Arrays عند إعادة تعريفها.
- جملة ReDim

- المصفوفات متغيرة الحجمDynamic Array

- في بعض الأحيان، لا نعرف مسبقاً حجم المصفوفة التي سنستخدمها في البرنامج بالضبط، وقد نريد تغيير حجم المصفوفة أثناء تشغيل البرنامج، هنا سنحتاج إلى المصفوفات ذات الحجم المتغير Dynamic حيث يمكننا تغيير حجمها في أي وقت.

تعتبر المصفوفات متغيرة الحجم أحد مميزات الفيجوال بيسك، وهي تساعد في تنظيم الذاكرة بكفاءة. فمثلاً، يمكن استخدام مصفوفة كبيرة لوقت قصير ثم إعادة تحجيمها لتحرير مساحة من الذاكرة عندما لا نحتاجها. وهذا من شأنه تسريع المعالجة.

ولإنشاء Dynamic Array نتبع التالي :

1. نعلن عنها بأحد أوامر الإعلان Public or Private or Dim or Staticونجعلها ديناميكية بعدم كتابة أي رقم في الأقواس كما يوضح المثال التالي

```
Dim AnyArray( )
```

2. نعيد الإعلان عنها مع تحديد عدد العناصر باستخدام جملة ReDim كما في المثال التالي

```
ReDim AnyArray( x+1 )
```

ملاحظات

- يمكن لجملة ReDim الظهور فقط في الـ Procedure وبعكس جملتي Dim, & Staticفإن جملة ReDim قابلة للتنفيذ Executable .
- تستخدم جملة ReDim نفس الصيغة Syntax المستخدم مع مصـــفوفات الحجم الثابت Fixed Array .
- كل جملة من جمل ReDim يمكنها تغيير عدد العناصر بالإضـــافة إلى الحـــد الأعلى والحد الأدنى لكل بعد للمصفوفة، ومع ذلك فـــإن عـــدد الأبعـــاد في المصفوفة لا يمكن تغييره.

مثال

لانشاء مصفوفة M كمصفوفة متغيرة الحجم، نعلن عنها أولاً في المـــديول Module على النحو التالي

```
Dim M( ) As Integer
```

ثم نعيد الإعلان عنها داخل الأجراء Procedure على النحو التالي

```
Sub ANY_NAME()
    ReDim M(9, 15)
End Sub
```

سندرس ما يلي:

١. الإعلان عن مصفوفات الحجم الثابت Fixed-size Array
٢. تحديد الحدين الأعلى والأدنى للمصفوفة Upper bound & Lower bound
٣. المصفوفات متعددة الأبعاد Multi-Dimensional Arrays
٤. جملة Option Base
٥. استخدام الدورات Loops لمعالجة المصفوفات.

الإعلان عن مصفوفات الحجم الثابت Fixed-size Array

هناك ثلاث طرق للإعلان عن هذا النوع تعتمد على النطاق **scope** الذي سوف تغطيه المصفوفة

- للإعلان عن مصفوفة عامة **Public Array** تستخدم جملة **Public** في قسم الإعلان **Declaration Section** للوحدة **Module**
- للإعلان عن **Module-Level Array** تستخدم جملة **Private** في قسم الإعلان الخاص بالأجراء **Procedure**
- للإعلان عن مصفوفة محلية **Local Array** تستخدم جملة **Dim** في الأجراء **Procedure**

تحديد الحدين الأعلى والأدنى للمصفوفة Upper bound & Lower bound

- عند الإعلان عن مصفوفة، يكتب الحد الأعلى بعد الاسم وبين الأقواس.
- لا يمكن أن يزيد الحد الأعلى عن نطاق نوع المتغير **Long Data Type**
- الحد الأدنى الإفتراضي **Default** هو الصفر.
- لتحديد الحد الأدنى، ينبغي كتابته صراحة باستخدام كلمة **To** ، كما سنرى في الأمثلة التالية.

مثال

إعلان عن مصفوفة **StudentId** يمكن وضعه في جزء الأعلان الخاص بالـ **Module** أو داخل الأجراء **Procedure**

Dim StudentId (1000) As Integer

وبذلك يكون عدد عناصر هذه المصفوفة ١٠٠١ عنصر. ولجعلها مصفوفة عامة، نستبدل **Dim**بـ **Public** كما يلي:

```
Public StudentId (1000) As Integer
```

ولتحديد الحد الأدنى لهذه المصفوفة بـ ١ وبالتالي يصبح عدد عناصرها ١٠٠٠ عنصر فقط، نكتب To كالتالي

```
Public StudentId (1 To 1000) As Integer
```

مثال

الإعلان عن مصفوفة بها ٥١ عنصر مرقمة من ٠ إلى ٥٠ من دون تحديد نوع بيانات عناصرها، سيعتبر النوع **Variants** مباشرة

```
Dim DayArray (50)
```

مثال

الإعلان عن المتغير **BirthDate** على أنه مصفوفة من التواريخ المفهرسة من ١ حتى ١٠

```
Dim BirthDate (1 To 10) As Date
```

المصفوفات متعددة الأبعاد Multi-Dimensional Arrays

يقصد بكلمة بُعد هو عدد الدلائل **Indexes** المستخدمة للرجوع إلى عنصر في المصفوفة. تستخدم المصفوفات متعددة الأبعاد لتخزين بيانات مرتبطة ببعضها البعض.

مثال :

تخزين كل نقطة **Pixel** موجودة على شاشة الحاسب نحتاج لتخزين إحداثياقما **x & y** وهذا ممكن باستخدام مصفوفة ذات بعدين.

يمكن معرفة أبعاد مصفوفة بالنظر إلى تعريفها؛ حيث يتم الإعلان عن مصفوفة متعــددة الأبعاد بذكر قيمة لأبعادها بعد اسمها مباشرة وداخل الأقواس، يتم فصل قيمة كل بعـــد بالفاصلة، سواء كانت ذات بعدين أو ثلاثة أبعاد كما ستوضح الأمثلة الآتية

- **Static A(3, 4) As Double**
- **Static A(1 To 10, 1 To 10) As Integer**

Dim B (3, 1 To 10, 1 To 15)

ملاحظة

- عدد عناصر المصفوفة متعددة الأبعاد يمكن معرفته بضرب قيم الأبعاد ببعضها.
- عند إضافة أبعاد المصفوفة فإن مساحة التخزين المطلوبة سوف تزيد زيادة كبيرة ولذلك ينبغي الاحتراس وتفادي استخدام النوع **Variant** قدر الإمكان لما يتطلبه من مساحة تخزينية كبيرة.

- جملة **Option Base**

- تستخدم في مستوى المديول **Module Level** للإعلان عن الحد الأدنى المفتـــرض لدلائل **subscripts** المصفوفات التي سوف تظهر في الـــ**Module**.

الصيغة **Syntax**

Option Base { 0 | 1 }

مع ملاحظة

- القيمة الافتراضية هي صفر دوماً، ولذلك لا داعي لكتابة Option Base 0إلا إذا كنّا نريد تذكير قارئ الكود بذلك
- إذا استخدمت هذه الجملة ينبغي ظهورها مرة واحدة وقبل أي Procedure وقبل أي جملة إعلان عن مصفوفات.
- جملة Option Base لها تأثير فقط على الحد الأدنى للمصفوفات في الـ Module التي توجد به الجملة.

مثال

يوضح المثال التالي كيفية تحديد القيمة الافتراضية للحد الأدنى لدلائل المصفوفات، ثم يوضح كيفية استخدام دالة الإعلام عن الحد الأدنى لمصفوفة LBound

```
Option Base 1
Dim Lower
Dim MyArray(20), TwoArray(3, 4)
Dim ZeroArray(0 To 5)
'Use LBound function to test lower bound of arrays.
Lower = LBound (MyArray)          'Returns 1
Lower = LBound (TowArray, 2)      'Returns 1
Lower = LBound (ZeroArray)        'Returns 0
```

استخدام الدورات Loops لمعالجة المصفوفات

يمكن بكفاءة وسهولة معالجة المصفوفات باستخدام الدورات Loops وخصوصاً المصفوفات متعددة الأبعاد حيث يتم معالجتها باستخدام Loops متداخلة.

مثال

الجمل التالية سوف تعطي قيمة لكل عنصر من عناصر المصفوفة A هذه القيمة تكون مرتبطة بموقع العنصر داخل المصفوفة كما سنرى

```
Dim I As Integer, J As Integer
Static A(1 To 10, 1 To 10) As Integer
      For I=1 To 10
         For J=1 To 10
          A(I, J)= I*10*J
   Next J
Next I
```

- حفظ محتويات المصفوفة متغيرة الحجم عند إعادة تعريفها

- يلاحظ أنه تيتم محو جميع القيم المخزنة في المصفوفة كل مرة يعاد فيهـا تنفيـذ جملـة **ReDim**ويجعل الفيجوال بيسك القيم كالتالي

Variant Array	في حالة ال____	**Empty Value**
Numeric Array	في حالة ال____	**Zero**
String Array	في حالة ال____	**Zero-Length String**
Array of objects	في حالة ال____	**Nothing**

وهذا مفيد عندما نريد تجهيز المصفوفة لبيانات جديدة أو عندما نريـد اختـزال حجـم المصفوفة لتأخذ أقل مساحة ممكنة في الذاكرة. ولكن أحياناً نريد تغيير حجم المصـفوفة دون فقد بياناها.

يمكننا فعل ذلك باستخدام جملة **ReDim** مع كلمة **Preserve** وتعني" احفظ".

مثال

الكود التالي يستخدم جملة ReDim لتخصيص، ثم إعادة تخصيص مســاحة تخزينيــة لمصفوفة متغيرة الحجم، مع افتراض أن الأساس Option Base يساوي ١

```
Dim MyArray ( ) As Integer      'declare Dynamic Array
  ReDim MyArray( 5 )            'allocate 5 elements
    For I = 0 To 5              'loop 5 times
      MyArray(I) = I            'initialize array elements
    Next I


  ReDim MyArray( 10 )           'allocate 10 elements
    For I = 0 To 10             'loop 10 times
          MyArray(I) = I        'initialize array elements
    Next I
```

الجملة التالية تغير حجم المصفوفة ولكنها لا تمحو العناصر الموجودة بها

```
ReDim Preserve MyArray( 10 )
```

والآن يمكننا كتابة ملخص متكامل لجملة ReDim .

جملة ReDim

تستخدم في مستوى الأجراء Procedure لإعادة تخصيص allocates مســاحة تخزينية storage space لمصفوفة متغيرة الحجم Dynamic array .

صيغته Syntax

ReDim [Preserve] *varname*(subscripts) [As type] [, varname (subscripts) [As type]]

حيث أن

Preserve	اختيارية، وتستخدم لحفظ البيانات الموجودة في المصفوفة عند تغيير حجم آخر بعد فيها.
varname	اسم المتغير اسم المصفوفة.
subscripts	أبعاد المصفوفة عددهم على الأكثر ٦٠ وتستخدم الشكل التالي [*lower* To] *upper* [, [*lower* To] *upper*] والقيمة الأدنى تحددها جملة **Option Base** وإذا لم توجد، يكون الحد الأدنى هو الافتراضي صفر.
type	اختيارية، وتحدد نوع بيان عناصر المصفوفة. وقد تكون أي من الأنواع التالية **Byte, Boolean, Integer, Long, Single, Double, Currency, Decimal, Date, Object, String, Variant, User-defined.**
[, varname (subscripts) [As type]]	اختيارية، لإعادة تحجيم مصفوفة أخرى.

ملاحظات هامة

- جميع ما ذكر في الصيغة داخل قوسين مربعين [ ] يعتبر اختياري يمكن الاستغناء عنه حين عدم الحاجة إليه.
- تستخدم جملة **ReDim** لتحجيم أو إعادة تحجيم مصفوفة متغيرة الحجم **Dynamic Array**والتي بالفعل قد أعلن عنها مسبقاً باستخدام أي من الجمل **Dim, Private, Public** مع أقواس فارغة أي بدون ذكر الأبعاد
- يمكن تكرار استخدام جملة **ReDim** لتغيير عدد العناصر والأبعاد لمصفوفة، ومع ذلك لا يمكن الإعلان عن مصفوفة بنوع معين من البيانات ثم إعادة

تعريفها لاحقاً مع تغيير نوع البيان لنوع آخر إلا إذا كانت المصفوفة محتواه في variant

- إذا كانت المصفوفة محتواه في **variant** فإن نوع بيان العناصر يمكن أن يتغير باستخدام المقطع **As Type** إلا إذا استخدمنا كلمة **Preserve** ففي هذه الحالة لا يسمح بتغييرات.
- إذا استخدمنا كلمة **Preserve** يمكن فقط تحجيم البعد الأخير للمصفوفة ولا يمكن تغيير عدد الأبعاد على الإطلاق.
- إذا كان للمصفوفة بعد واحد فيمكن إعادة تحجيم هذا البعد لأنه البعد الأخير والوحيد بالمصفوفة.
- وإذا كان للمصفوفة بعدين أو أكثر فيمكن فقط تغيير حجم البعد الأخير مع الاحتفاظ بمحتويات المصفوفة.
- عندما نستخدم **Preserve** يمكن تغيير حجم المصفوفة بتغيير الحد الأعلى بينما ينتج لدينا خطأ حين تغيير الحد الأدنى.
- إذا اعددنا مصفوفة أصغر مما كانت فإن بيانات العناصر المخزنة سوف تفقد.
- إذا مررنا **pass** مصفوفة إلى أجراء **Procedure** بالـ **Reference** فإنه لا يمكن تغيير أبعادها من داخل الأجراء.

تحذير

جملة **ReDim** ستعمل وكأنها جملة إعلان إذا كان المتغير المصفوفة التي تعلن عنه غير موجود على مستوى الـ **Procedure** أو الـ **Module** وإذا كان هناك متغير آخر بنفس الاسم قد أنشئ بعد ذلك وحتى لو كان في النطاق ككل **Scope** ؛ فإن **ReDim** سوف ترجع للمتغير الأخير ولن يتسبب عن ذلك خطأ في الترجمة **Compilation error** حتى ولو كانت جملة **Option Explicit** فعّالة. وبذلك لن يدرك المبرمج أنه هناك خطأ بالكود.

ولتفادي هذا التعارض لا ينبغي استخدام جملة **ReDim** كجملة إعلان بدلاً من **Dim** مثلاً، ولكن نستخدمها فقط لإعادة تعريف حجم المصفوفة.

مثال

يبين المثال التالي كيف يمكن زيادة حجم البعد الأخير للمصفوفة متغيرة الحجم بدون محو البيانات الموجودة فيها

```
ReDim X(10,10,10)
ReDim Preserve X(10,10,15)
```

مثال

لإدخال قائمة من الأعداد غير معلومة العدد مسبقاً أي وقت كتابة الكود ويسأل عن عددها عند تشغيل الكود

```
Dim X() As Integer
N = InputBox("أدخل عدد العناصر من فضلك")
ReDim X(N)
For I = 1 To N
     X(I) =  InputBox("أدخل عنصر من القائمة")
Next I
```

مثال

لإدخال قائمة من الأعداد غير معلومة العدد مسبقاً أي وقت كتابة الكود code ، ولا يسأل عن عددها عند تشغيل الكود ولكن يقوم المستخدم بإدخال صفر أو أي قيمة متفق عليها عند الأنتهاء من ادخال عناصر المصفوفة

```
Dim X()
Do
   prompt="أدخل عنصراً من القائمة وفي النهاية أدخل صفر"
   DummyVariable = InputBox(prompt)
   IF DummyVariable <> 0 Then
          UpperLimit = UpperLimit + 1
```

```
            ReDim Preserve X(UpperLimit)
            X(UpperLimit) = DummyVariable
    Else
            Exit Do
    End IF
Loop
```

الفصل السابع

بعض الدوال الرياضية فى فيجوال بيسك

دالة القيمة المطلقة Abs :

تحدد القيمة المطلقة لأي رقم وترجعه من نفس نوع البيانات المعطى للدالة والمقصود بالقيمة المطلقة هي قيمة العدد بدون إشارة فالقيمة المطلقة ل -١٣ مثلا هي ١٣ وهكذا، فمثلا لو كتبنا الكود التالي :

```
MyNumber = Abs(-45.6)
Text1.Text = MyNumber
```

فإن نتيجة تنفيذ الدالة هي **MyNumber = 45.6**

لاحظ أن القيمة المدخلة للدالة لابد أن تكون عدد أو تعبير عددي فإذا كانت القيمة المدخلة للدالة **Null** ستكون النتيجة **Null** وإذا كانت القيمة المدخلة للدالة متغير فارغ أو لم يتم تعيين قيمة له ستكون النتيجة (٠).

دالة الجزر التربيعى Sqr :

تستخدم هذه الدالة في تحديد الجذر التربيعي لرقم معين وتأخذ الصورة العامة التالية.

```
MyNumber = Sqr(10)
Text1.Text = MyNumber
```

فإن نتيجة تنفيذ الدالة هي **MyNumber = 3.1622776**

دالة اللوغاريتم الطبيعى Log :

تستخدم هذه الدالة في تحديد قيمة اللوغاريتم العشري لرقم وتأخذ الصورة العامة التالية

```
MyNumber = Log (20)
Text1.Text = MyNumber
```

فإن نتيجة تنفيذ الدالة هي **MyNumber = 2.9957327**

دالة الأس **Exp** :

تستخدم هذه الدالة في تحديد القيمة (e) وهي قاعدة اللوغاريتم الطبيعي مرفوعة بقوة الرقم الذي تتضمنه حيث (e) تساوي تقريبا 2.7182818 وتأخذ الصورة العامة التالية:

MyNumber = Exp (رقم)

دالة الأرقام العشوائية **Rnd**

تستخدم هذا الدالة في توليد أرقام عشوائية تقع ما بين الصفر و واحد بحد أقصـــي ١٥ رقما عشريا وتأخذ الصورة العامة التالية:

MyNumber = Rnd (عدد)

فمثلا الدالة **Rnd(10)** قد تعطي رقما مثل ٠,٧٠٥٥٤٧٥ وعند تشغيل الدالة مرة أخري ينتج رقما آخر مثل **0.533424** وهكذا

دالة العدد الصحيح **Int**

تستخدم هذه الدالة لحساب الجزء الصحيح فقط من رقم يشتمل علي أرقام صـــحيحة وعشرية أو بعبارة أخر لحذف الأرقام العشرية الموجودة بعد العلامة العشرية بدون تقريب وتأخذ الصورة التالية

MyNumber = Int (332.54)

فإن نتيجة تنفيذ الدالة هي **MyNumber = 332**

دالة مقلوب ظل الزاوية **Atn** :

تستخدم هذه الدالة في حساب مقلوب ظل الزاوية "ظتا" للرقم الذي تشتمل علية مقدار بالتقدير الدائري وتأخذ الصورة العامة التالية:

MyNumber = Atn (رقم)

دالة ظل الزاوية Tan :

تستخدم هذه الدالة في تحديد قيمة ظل زاوية معينة وتأخذ الصورة العامة التالية :

MyNumber = Tan (رقم)

دالة جيب تمام الزاوية Cos :

وتستخدم هذه الدالة في تحديد قيمة جيب تمام الزاوية معينة وتأخذ الصورة العامة التالية

MyNumber = Cos (رقم)

دالة جيب الزاوية Sin :

تستخدم هذه الدالة في تحديد قيمة جيب زاوية معينة وتأخذ الصورة العامة التالية:

MyNumber = Sin (رقم)

دوال التعامل مع البيانات الحرفية

دالة الحرف المناظر للرقم chr:

تقوم الدالة chr بأخذ قيمة بين ٠ و ٢٥٥ وتعيد الحرف الممثل لهذه القيمـــة في جدول رموز ASCII، على سبيل المثال العبارة التالية:

Hi، I'm "mohamed"

ستجد أنك لا تستطيع كتابتها بهذا الشكل:

myText = "Hi & "،vbCrLf" & I'm "mohamed ""

لأن البرنامج سيعتقد بأن نهاية السلسة النصية السابقة هي عند علامات الاقتبـــاس التي تقع مباشرة قبل كلمة mohamed وستظهر لك رسالة خطأ، لذلك فإننا

نلجأ لاستخدام الدالة **chr** حيث أن رمز علامة الاقتباس المزدوجة في جــدول **ASCII** هو ٣٤، فتكون الصياغة الصحيحة للجملة السابقة كالتالي:

```
myText = "Hi & "،vbCrLf" & I'm & " chr(34
)"mohamed & "chr(34)
```

دالة عدد حروف الجملة **Len**:

ستجد حاجتك في كثير من الأحيان لمعرفة طول السلسلة النصية (عدد الأحرف)، ولعمل ذلك استخدم الدالة **Len**، مرر إليها النص وستعيد لك عدد الحروف.

```
myLength = Len("mohamed")
```

أى ٧ حروف.

دالة ايجاد مكان احد الحروف **InStr**

```
InStr([start ,]string1, string2[, compare])
```

يمكنك بواسطة هذه الدالة معرفة أول مكان يظهر فيه نص ما ضمن نص آخر أكبر منه.

الوسيطة الأولى هي وسيطة اختيارية تحدد مكان بدء البحث، أما الوسيطة الثانيــة **string1** فتحدد السلسلة النصية التي سيتم البحث فيها، والوســيطة الثالثــة **string2** تحدد السلسلة النصية التي سيتم البحث عنها في السلسلة الأولى، أما الوسيطة الأخيرة فهي اختيارية أيضا وتحدد نوع المقارنة التي يجب إجرائهــا وهــي تأخذ أحد الثوابت التالية:

- vbBinaryCompare – ٠
- vbTextCompare – ١

والفرق بينهما هو أن الأولى تراعي حالة الأحرف والثاني لا تراعي حالة الأحرف.

في المثال التالي الدالة **i** ستحتوي على القيمة ١:

```
i = InStr("mohamed","m")
```

وأما المثال التالي فستحتوي **i** فيه على القيمة ٦:

```
i = InStr(2,"mohamed","m" )
```

حيث أن الدالة في المثال السابق ستبحث عن الحرف **m** بادئة من الحرف الثاني ولذلك فهي لن تجد الحرف الأول.

دالة تحويل القيم الى حروف **str$** **Str(number)**

قد تبدو الدالة **str** متشابهة مع الدالة **chr**، إلا أنها تؤدي وظيفة عكسها تماما، فهي تحول الأرقام إلى سلاسل نصية، وهي تفيد مثلا في حال أردت أن تقوم بالتحام بين رقمين فتقوم بتحويل كل منهما إلى سلسلة نصية وتطبق بينهما جمع السلاسل (&) الذي يختلف عن جمع الأرقام وبالتالي تحصل على سلسلة جديدة يمكنك أن تحولها إلى رقم من جديد، على سبيل المثال الدالة **myNumber** تحتوي على القيمة ١٢٣٤٥٦.

```
myNumber = Str(123 & (Str(456)))
```

ستواجهك مشكلة في الكود السابق حيث يقوم فيجوال بيسيك بإضافة مسافة قبل كل سلسلة نصية تنتج من الدالة السابقة.

التحوير

يمكنك عمل الكثير من التغييرات والتحويرات في السلاسل النصية، الأمر يشبه تطبيق الفلاتر على هذه النصوص وتوجد عديد من هذه الفلاتر مدمجة في فيجوال بيسيك، وفيما يلي أهمها:

دالة الحروف صغيرة **LCase** **LCase(string)**:

تعيد الدالة **LCase** نسخة من النص **string** تكون فيه جميع الحروف صغيرة **Lowercase** المتغيرة **myText** في المثال التالي ستحتوي على عبارة **it works**:

```
myText = "It Works"
myText = LCase(myText)
```

دالة الحروف كبيرة UCase (UCase(string:

نفس الدالة السابقة لكنها تقوم بتحويل الأحرف إلى حروف كبيرة Uppercase.

دالة عكس الترتيب StrReverse (StrReverse(string:

تعيد السلسلة string معكوسة، أي تبدأ من البداية وتنتهي من النهايـــة، المـــتغيرة myText في المثال التالي ستحتوي على العبارة emoclew :

```
myText = strReverse("welcome")
```

دالة الاستبدال Replace:

Replace(string1, string2, string3, start, count[, compare])

وتقوم باستبدال النص string2 بالنص string3 ضمن السلســـلة string1، أي أنها تبحث في النص string1 عن النص string2 وعندما تجده فإنها تحذفه منه وتضع مكانه string3، ويمكن تحديد نقطة بداية البحث بالوسيطة start، وعدد مرات الاستبدال القصوى بالوسيطة count، ضع القيمة -١ لاســـتبدال الكـــل، وطريقة المقارنة بالوسيطة compare كما ذكر سابقا.

المتغيرة myText في المثال التالي ستحتوي على القيمة Hi Everyone

```
myText = Replace("Welcome Everyone,",
"Welcome", "Hi", 1, -1)
```

يمكنك أن تأخذ أجزاء محددة من سلسلة نصية بعدة طرق مختلفة، يمكنك مـثلا أن تحصل على الأحرف الثلاثة الأولى من بداية السلسلة النصية أو الأربعة الأخيرة أو خمسة أحرف بدءا من الحرف السادس وهكذا.

دالة الاستخلاص من البداية Left$ Left(string, length)

حيث تقوم بوضع سلسلة نصية **string** وتحدد الجزء الذي تريد اقتطاعه من بداية السلسلة **length** ، والتعبير بكلمة من بداية السلسلة أدق من يسار السلسلة لأن هذا قد يحدث اشتباها في السلاسل النصية للغات التي تكتب من اليمين إلى اليسار مثل العربية، هنا تعيد السلسلة العدد المحدد من الحروف من اليمين أي أنها لا تهتم لاتجاه ظهور أحرف السلسة وإنما اتجاه تخزينها، وللتخلص من هذه المشكلة سنقول بداية السلسلة .

دالة الاستخلاص من النهاية Right$:

مطابقة للدالة **Left** في كل شيء، إلا أنها تأخذ العدد المحدد من الحروف من نهاية السلسلة .

دالة الاستخلاص من الداخل Mid$ Mid (string, start[,length])

تعيد الدالة **Mid** عددا من الأحرف قدره **length** بدءا من حرف معين هو **start**، في سلسلة نصية **string** لاحظ أيضا أن الوسيطة **length** اختيارية وإذا لم تمرر بها أي قيمة فإن الدالة ستعيد الأحرف إلى نهاية السلسلة .

في المثال التالي ستحتوي المتغير myText على العبارة 'm m' لاحظ اعتبـــار المسافة حرف :

```
myText = Mid("I'm mohamed", 2, 4)
```

أما في المثال التالي فستحتوي على الكلمة mohamed:

```
myText = Mid("I'm mohamed",4, 5)
```

الدالة Mid

تقوم الدالة Mid باستبدال مقطع محدد من النص بنص آخر، وهو يكتـــب في صورة مشابهة جدا لطريقة كتابة دالة Mid ولكن توضع بعده علامة مســـاواة وبعدها العبارة الجديدة، في المثال التالي ستحتوي المتغيرة myText على القيمة :I'm mohamed

```
name = "mubarmej"
myText = "I'm name"
Mid(myText, 5) = name
```

حيث سيتم حذف الجزء المحدد بالخاصية Mid ويوضع الجزء الذي بعد علامـــة المساواة في مكان الجزء المحذوف .

عمل ملف تنفيذي للبرنامج: أختر make project1.Exe من قائمة file ثم حدد الدليل الذي تريد نسخ الملف فيه وسمي الملف أصبح لديك برنامج يعمل بمفرده دون تشغيل فيجول بيسيك لاكن يجب الانتباه إلى أن هذا البرنـــامج لا يعمل setup على جهاز لا يحتوي على فيجول بيسيك ولكي تجعل برنامجك يعمل على جميع الأجهزة يجب عمل حزمة برامج.

دوال التعامل مع التاريخ

أهم الدوال للتعامل مع التاريخ والوقت في الفيجوال بيسك

أولا: نوع بيانات التاريخ

يخزن التاريخ داخل الحاسب على هيئة ٨ – byte حيث يخزن كرقم ذا فاصلة عشرية عائمة Floating حيث يعرض التاريخ من المدى ١ يناير سنة ١٠٠ إلى ٣١ ديسمبر ٩٩٩٩ ومدى الوقت يكون من ٠٠,٠٠:٠٠:٠ إلى ٢٣:٥٩:٥٩ حيث يتم وضع التاريخ بين العلامتين # # والتاريخ الافتراضي المخزن بالجهاز هو التاريخ ذو الصيغة القصيرة كذلك الوقت يعرض بالصيغة القصيرة حسب نظام ١٢ ساعة أو ٢٤ ساعة عندما يتم تحويل أي أرقام إلى تاريخ فإن القيم العشرية على يمين الفاصلة العشرية تعطي التاريخ والأرقام على يسار الفاصلة العشرية تعطي الوقت

ثانيا:دوال التاريخ

١ – دالة معرفة التاريخ الحالي Date

وتستخدم لعرض التاريخ الحالي المخزن في نظام التشغيل وهو الشكل القصير

dd/mm/yyyy

الشكل العام للدالة Date ( )

مثال

اضغط على زرار عرض واكتب الأمر التالي

```
Text1 = date
```

٢ – أمر التاريخ date

حيث تستخدم لإعادة ضبط تاريخ الجهاز إلى تاريخ معين محدد

الشكل العام للأمر Date = date

مثال : اكتب الكود التالي في زرار عرض

```
Dim mydate
mydate = #12/10/1995#
```

```
Date = mydate
Text1 = Date
```

٣ – دالة معرفة الوقت الحالي time

تستخدم لعرض الوقت الحالي المخزن في نظام التشغيل وهو الشكل القصير
HH:MM:SS

الشكل العام للدالة () time

مثال : اضغط على زرار عرض واكتب الأمر التالية

```
Text3 = date
```

٤ – أمر التوقيت Timer

يستخدم لمعرفة قيمة الثواني للوقت الحالي ويمكن استخدامها كعداد

الشكل العام للأمر timer

مثال :يعرض المثال التالي نموذج لإيقاف التطبيق لمدة خمس ثواني مع السماح بالعمل على التطبيقات الأخرى باستخدام أمر Doevents. اضغط زرار عداد مرتين واكتب الكود التالي :

```
Dim PauseTime, Start, Finish, TotalTime
If (MsgBox("Press Yes to pause for 5 seconds",
4)) = vbYes Then
 PauseTime = 5
 Start = Timer
 Do While Timer < Start + PauseTime
   DoEvents
 Loop
 Finish = Timer
 TotalTime = Finish - Start
```

```
 MsgBox "Paused for " & TotalTime & " seconds"
Else
 End
End If
```

٥ – دالة الوقت والتاريخ Now

لعرض الوقت والتاريخ الحاليين

الشكل العام للأمر Now

مثال : أضف السطر التالي للكود في زرار عرض

```
Text2 = now
```

العمليات على التاريخ

١ – دالة الإضافة إلى تاريخ dateadd

تستخدم لإضافة أو طرح قيمة محددة إلى تاريخ محدد

الشكل العام للدالة

```
DateAdd(interval, number, date)
```

شرح بارامترات الدالة

١ – **interval** : وهي القيمة المراد إضافتها إلى التاريخ المعطي ولها عدة أشكال مشروحة في الجدول التالي

م	القيمة	الشرح
YYYY	Year سنة	حيث يضيف سنة إلى التاريخ المعطى
q	Quarter ربع سنة	حيث يضيف ثلاث شهور إلى التاريخ المعطى
m	Month شهر	حيث يضيف شهر إلى التاريخ المعطى
y	Day of year يوم من السنة	حيث يضيف يوم إلى التاريخ المعطى
d	Day يوم	حيث يضيف يوم إلى التاريخ المعطى
w	Weekday يوم اسبوعي	حيث يضيف يوم إلى التاريخ المعطى
ww	Week اسبوع	حيث يضيف اسبوع كامل إلى التاريخ المعطى
h	Hour ساعة	حيث يضيف ساعة إلى التاريخ المعطى
n	Minute دقيقة	حيث يضيف دقائق إلى التاريخ المعطى
s	Second ثانية	حيث يضيف ثواني إلى التاريخ المعطى

٢ – البارامتر **Number**

وهو بارامتر العدد المطلوب إضافته ويمكن إن يكون موجب فيضيف إلى التاريخ أو سالب فيطرح من التاريخ .

٣ – البارامتر **date**

وهو التاريخ المراد الإضافة أو الحذف منه

مثال: اضغط على زرار إضافة إلى التاريخ ثم اكتب الكود التالي :

```
Dim FirstDate As Date   ' Declare variables.
Dim IntervalType As String
Dim Number As Integer
Dim Msg
IntervalType = "m"   ' "m" الأشهر
FirstDate = InputBox("Enter a date") 'ادخل أي تاريخ
Number = InputBox("Enter number of months to
add") 'بالموجب أو بالسالب أضف القيمة
Msg = "New date: " & DateAdd(IntervalType,
Number, FirstDate)
Text1 = Msg
```

٢ – الدالة datediff

وهي دالة تعطي الفرق بين تاريخين محددين وتعود بقيمة صحيحة طويلة .

الشكل العام للأمر

```
DateDiff(interval, date1, date2[, firstdayofweek ،]
firstweekofyear]])
```

شرح بارامترات الأمر

١ – البارامتر **Interval**

١ – **interval** : وهي القيمة المراد عرضها كناتج الطرح ولها عدة إشكال مشروحة في الجدول السابق

٢ – **date١** التاريخ الأول

٣ – **date٢** التاريخ الثاني

٤ – البارامتر **firstdayofweek** وهو اختياري وهو لتحديد يوم البدء للأسبوع مثل المسلمين بداية الأسبوع لهم السبت .... وهكذا وله القيم التالية

م	القيمة	الشرح
vbUseSystem	0	كما هو مستخدم في النظام
vbSunday	1	الأحد
vbMonday	2	الاثنين
vbTuesday	3	الثلاثاء
vbWednesday	4	الأربعاء
vbThursday	5	الخميس
vbFriday	6	الجمعة
vbSaturday	7	السبت

٥ – البارامتر **firstweekofyear** وهو اختياري وهو لتحديد الأسبوع الأول في العام وهو غير مهم بالنسبة لاستخدامات الدالة

مثال: اضغط على طرح تاريخين مرتين واكتب الكود التالي والذي يطرح التاريخ المعطى من التاريخ الحالي ويعطي الناتج بالأيام

```
Dim TheDate As Date   ' Declare variables.
Dim Msg
TheDate = InputBox("Enter a date")
Msg = "Days from today: " & DateDiff("d", Now,
TheDate)
Text1 = Msg
```

٣ – دالة Dtaepart

تستخدم لإرجاع قيمة صحيحة من تاريخ محدد

الشكل العام للدالة

```
DatePart(interval, date[,firstdayofweek[,
firstweekofyear]])
```

وقد تم شرح البارامترات في الدوال السابقة

مثال: اضغط على زرار إرجاع قيمة من تاريخ واكتب الكود التالي لإرجاع رقم الربع من التاريخ المعطى

```
Dim TheDate As Date   ' Declare variables.
Dim Msg
TheDate = InputBox("Enter a date:")
Msg = "Quarter: " & DatePart("q", TheDate)
Text1 =  Msg
```

٤ – دالة Datevalue

تستخدم لإرجاع التاريخ من القيمة المعطاة حيث تكون القيمة المعطاة قيمة نصية (لا أعني أنها حروف ولكن أرقام تقرأ على أنها نص كما تأتي الأرقام من مربع نص في الحقيقة نص وليست رقم)

الشكل العام للدالة **DateValue(date)**

حيث **date** هو التاريخ حيث يمكن إدخال التاريخ كل قيمة نصية بينها فاصلة

مثال : اضغط على زرار

```
Dim MyDate As Date
MyDate = DateValue("8,2005")   ' Return a date.
Text1 = MyDate
```

٥ – دالة dateserial

وهي دالة تعود بالتاريخ من قيم معطاة لليوم والشهر والسنة

الشكل العام للدالة

```
DateSerial(year, month, day)
```

شرح بارامترات الدالة

١ – **year** وهي السنة المعطاة وهو رقم يمكن أن يكون بين ١٠٠ إلى ٩٩٩٩

٢ – **month** وهو الشهر المعطى وهو أي رقم صحيح

٣ – **day** اليوم المعطى وهو أي رقم صحيح أيضا

يمكنك إجراء عمليات حسابية على الأرقام المدخلة مثل طرح ١٠ سنوات من العام المعطى وهكذا وكذلك يمكنك استخدام مربعات النصوص لوضع تاريخ محدد أو لإعطاء القيم

مثال: اضغط على زرار **datesrial** واكتب الكود التالي

```
Dim MyDate
MyDate = DateSerial(1969 - 10, 2, 12)
Text1 = MyDate
```

٦ – دالة timevalue

تستخدم لإرجاع الوقت (قيمة تعبر عن وقت) من القيمة المعطاة حيث تكون القيمة المعطاة قيمة نصية

الشكل العام للدالة

```
TimeValue(time)
```

حيث **time** هو الوقت المعطى حيث يمكن إدخال الوقت كقيمة نصية بينها نقطتين ويجب الالتزام بحجم كل قيمة بمعنى عدد الساعات لا يزيد عن ٢٤ ساعة (المجموع الكلي للوقت) وكذلك عدد الدقائق لا تزيد عن ٦٠ وكذلك عدد الثواني

مثال : اضغط على زرار **timvalue** واكتب:

```
Dim MyTime
MyTime = TimeValue("20:62:20 PM")
Text3 = MyTime
```

٧ – دالة timeserial

وهي دالة تعود بالوقت من قيم معطاة للساعات والدقائق والثواني
الشكل العام للدالة

```
TimeSerial(hour, minute, second)
```

شرح بارامترات الدالة

١ – hour وهي الساعات المعطاة وهو رقم يمكن أن يكون بين ٠ إلى ٢٣

٢ – minute وهو الشهر المعطى وهو أي رقم صحيح

٣ – second اليوم المعطى وهو أي رقم صحيح أيضا يمكنك إجراء عمليات حسابية على الأرقام المدخلة مثل طرح ١٠ ساعات من الساعات المعطاة وهكذا وكذلك يمكنك استخدام مربعات النصوص لوضع وقت محدد أو لإعطاء القيم وهي.

مثال : اضغط على زرار timeserial واكتب الكود التالي

```
Dim MyDate
MyDate = DateSerial(1969 - 10, 2, 12)
Text1 = MyDate
```

٨- دالة فحص التاريخ Isdate

وهذه الدالة تعود بقيمة Boolean لها القيمة true إذا كان ناتج الفحص تاريخ و false إذا كان ناتج الفحص ليس تاريخ

الشكل العام للدالة

```
IsDate(expression)
```

حيث expression هو التعبير المراد فحصه

مثال على الدالة

```
If IsDate(Text1.Text) Then
MsgBox "this Is  Date"
Else
MsgBox "this Is not Date"
End If
```

٩- دالة cdate

وهي تستخدم للتعامل مع القيمة المدخلة على أنها قيمة تاريخ بمعنى إمكانية التعامل مع

القيم المدخلة في مربع النص (نصوص) كتاريخ طبعا يجب أن تكون القيم على شكل تاريخ

الشكل العام للدالة

```
cdate( expression)
```

حيث expression هو التعبير المراد التعامل معها كتاريخ

مثال على الدالة

```
Dim MyDate, MyShortDate
MyDate = "10 5 2004"
MyShortDate = CDate(MyDate)
Text1.Text = MyShortDate
```

١٠-دالة fomat

وهذه الدالة مفيدة جدا عندما تريد أن تجعل التاريخ على شكل معين تريده

الشكل العام للدالة

```
Format(expression[, format[, firstdayofweek[,
firstweekofyear]]])
```

بارامترات الدالة

١ - expression وهي القيمة المدخلة المراد تشكيلها

٢ - format وهو الشكل الجديد المطلوب للقيمة المدخلة

٣ - firstdayofweek تم شرحها في الدرس السابق

٤ - firstweekofyear تم شرحها في الدرس السابق

مثال على الدالة

```
Dim MyTime, MyDate, MyStr
MyTime = #5:04:23 PM#
MyDate = #1/27/1993#
```

```
MyStr = Format(Time, "Long Time")
MyStr = Format(Date, "Long Date")
MyStr = Format(MyTime, "h:m:s")
MyStr = Format(MyTime, "hh:mm:ss AMPM")
MyStr = Format(MyDate, "dddd, mmm d yyyy")
MyStr = Format(23)
MyStr = Format(5459.4, "##,##0.00")
MyStr = Format(334.9, "###0.00")
MyStr = Format(5, "0.00%")
MyStr = Format("HELLO", "<")
MyStr = Format("This is it", ">")
```

الفصل الثامن

بعض المهارات فى فيجوال بيسك

مثال على برنامج بسيط:

يحتوى البرنامج على:

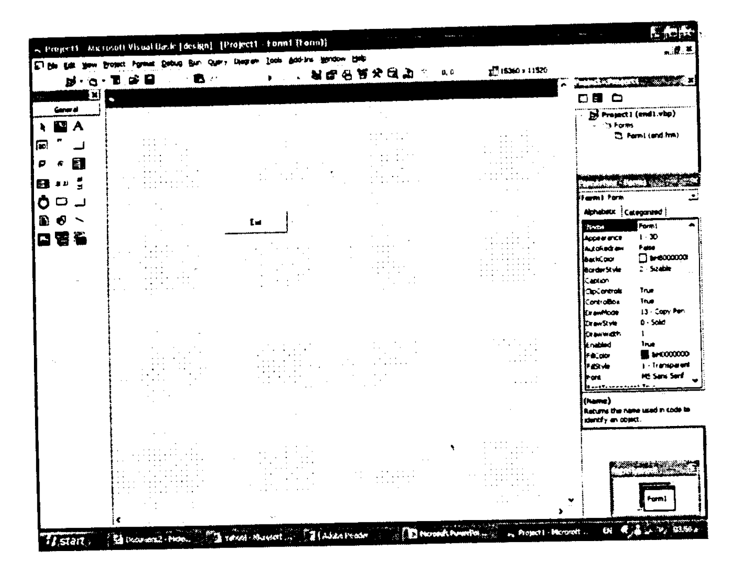
كائنين: كائن النموذج وكائن زرار أمر.

حدث: حدث الضغط على زرار الأمر.

إجراء: إغلاق البرنامج.

نفذ ما يلي:

- افتح مشروع جديد
- أضف زرار أمر للنموذج
- غير اسم زرار الأمر إلى **CmdEnd**
- غير الخاصية **Caption** للزرار لتكون **Exit**
- اضغط على **F5** ولاحظ ما يحدث.
- سيظهر لك النموذج كما يلي:



- لم يحدث شيء لأننا ل نضف الكود الخاص بالزرار.
- أرجع إلى النموذج واضغط ضغطتين على الزرار فتظهر لك نافذة الكود وبها الكود التالي

```
Private Sub CmdEnd_Click()

End Sub
```

- أضف بين السطرين الأمر End.
- اضغط على الزرار.
- لاحظ أن:

Twip - screen independent unit, 1440 twips to a logical inch
Pixel - picture element, number of pixels per inch varies with the screen

شرح أول كود

عند فتح نافذة كود لزرار مثلا نجد مكتوب الأسطر التالية

```
Private Sub Command1_Click()

End Sub
```

معنى الكلمات المستخدمة:

Private : تعنى أن الجزء التالي سيعمل من خلال النموذج فقط ولن يعمل من خلال اى نموذج آخر.

وإذا رغبت في جعله عاما اى من خلال اى جزء في البرنامج اكتب **Public** مكان **Private**.

Sub : تعنى أن الجزء المحصور بين **Sub** و **End Sub** عبارة عن برنامج فرعى متكامل.

Command1: تعنى أن الإجراء التالي خاص بالكائن الذي اسمه **Command1**.

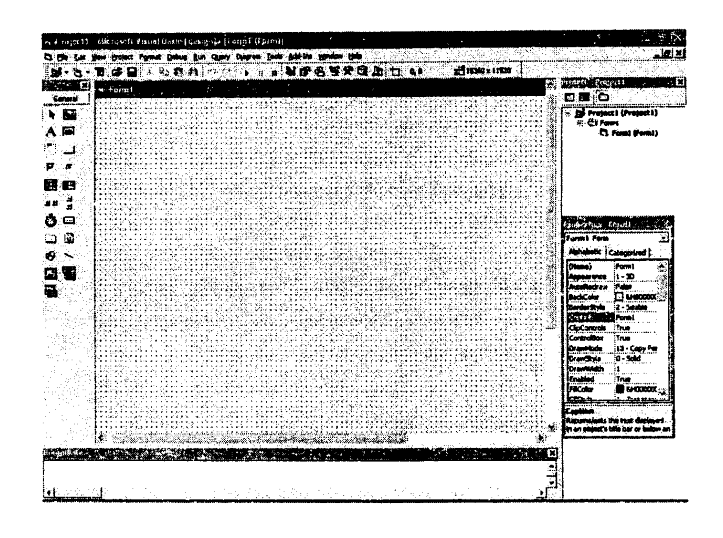
Click: تعنى أن هذا الإجراء سيتم تنفيذه في حالة الحدث **Click** ومجموع الكلمتين **Command1_Click()** تعنى أن هذا الإجراء يعمل من خلال الحدث **Click** فوق الزرار **Command1**.

End Sub: تعنى انتهاء الإجراء.

نافذة الفورم

نافذة الفورم هي النافذة الرئيسية للمشروع وستحتوى على كل الكائنات الأخرى التي تضيفها للنموذج.

لذالك فأن النموذج أهم كائن في المشروع.



والنموذج عبارة عن شكل رباعي يحتوى في أعلاه على شريط العنوان المكتوب فيه Form1 وبجواره أيقونة النموذج ويحتوى على عدد من النقاط، الفرق بين كل نقطة وأخرى ١٢٠ Twip .

بالضغط على الزرار الأيمن تظهر قائمة بعدد من إمكانيات الفورم مثل:

View Code : لإظهار نافذة الكود لكتابة الكود الخاص بالفورم.

Menu Editor: لإظهار نافذة محرر القوائم الذي يساعدك في إضافة قوائم إلى مشروعك.

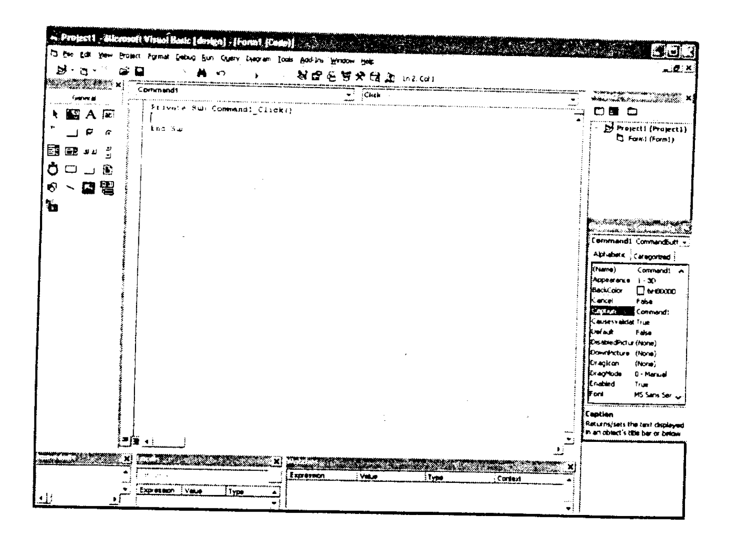
Lock Control : هذا الأمر يجعل جميع الأدوات في النموذج غير قابل للتحريك وتستخدمه بعد وصولك إلى الوضع النهائي لهذه الأدوات.

Past : للصق.

Properties: للانتقال إلى نافذة الخصائص.

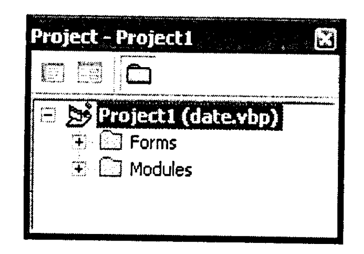
نافذة الكود

لاستعراض نافذة الكود أضغط على الكائن المراد إظهار الكود الذي يخصه.



تتكون نافذة الكود من شاشة فارغة لكتابة الكود يعلوها قسمين الأول لإظهار الكائن الذي تعمل عليه، والثاني لإظهار الأوامر التي يمكن تنفيذها عليه.

نافذة المشروع



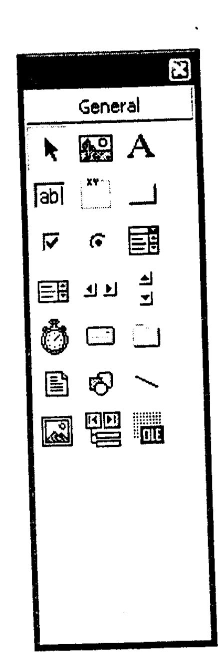
تحوى نافذة المشروع على كائنان الأول هو كائن الفهرس **Directory** والذي يحتوى على كل النماذج التي ستضيفها للمشروع ومن الممكن وجود **Directory** آخر عند إضافة **Module** أو **Class Module**...

والكائن الآخر هو من نوع النموذج **Form** وأسمه **Form1**.

تمكنك نافذة النماذج من تنفيذ عديد من الأوامر علـــى اى كـــائن مـــن الكائنـــات في مستكشف المشروع وذلك بالضغط على المفتاح الأيمن للفأرة واختيار حفظ أو محو....

صندوق الأدوات Tool Box

صندوق الأدوات من النوافذ الرئيسية في فيجوال بيسك حيث يحتوى علـــى كـــل الأدوات التي يمكن أن تضيفها للنموذج من صندوق صـــورة PictureBox أو ميقاتي Timer أو صندوق نص TextBox أو صندوق عنوان Label .



عناصر صندوق الأدوات

هذه العناصر تسمى أدوات Active x وهى أدوات جاهزة تساعدك في إنشـــاء البرامج.

Pointer: تعيد مؤشر الفأرة إلى السهم الطبيعي إذا كان المؤشـــر علـــى احـــد الأشكال الأخرى بغير الشكل الطبيعي.

PictureBox: لإضافة صور إلى البرنامج وإدماجها به.

Label: لإضافة عنوان إلى البرنامج أو نص في اى مكان على النموذج كما تستخدم لتوجيه المساعدة للمستخدم ولعرض النتائج التي نرغب في عـــدم تـــدخل المستخدم فيها.

Text Box: لإدخال بيانات مثل الأسماء والمرتبات، العمر...

Frame: لعمل إطار وتضمين بعض الأدوات داخل الإطار.

Command Box: زرار أمر يضغط عليه المستخدم لتنفيذ الأمر الموجود في كود هذا الزرار.

Check Box: مربعات اختيار تمكن المستخدم من تحديد اختياراته.

Option Button: دوائر اختيار تمكن المستخدم من تحديد اختيار وحيد من عدة اختيارات.

Compo Box: لإضافة قائمة منسدلة ليختار المستخدم منها إحدى القيم.

List Box: أداة قائمة List تشبه القائمة السابقة إلا أنها ليست منسدلة.

HScrollBar: عمود الانزلاق الأفقي ويســـتخدم لانـــزلاق الصـــور والكائنات الأكبر من الشاشة.

VScrollBar: عمود الانزلاق الرأسي ويســـتخدم لانـــزلاق الصـــور والكائنات الأكبر من الشاشة بشكل رأسي.

Timer: أداة الميقاتى وتنفذ عمل معين بصفة دورية كلما مر زمـــن معـــين تحدده Interval.

DriveListBox: قائمة تعرض أقسام الأسطوانة الصلبة والمرنة والســـى دى والفلاش.

DirListBox: قائمة تعرض المجلدات في مسار معين تحدده.

FileListBox: قائمة تعرض الملفات Files في مسار معين.

Shape: أداة رسم شكل.

Line: أداة رسم خط.

Image: أداة إضافة صورة.

Data: أداة تستخدم لربط البرنامج مع قاعدة بيانات.

OLE: أداة ربط وإدراج ملفات وبرامج ضمن برنامجك.

أزرار الاختيار Option Buttons

عند وضع أكثر من زرار اختيار فأنه باختيار احدهما فإن الباقي لا يعمل
فإذا وضعنا ٣ ازرار اختيار وسميناهم ١ و ٢ و ٣ ووضعنا ٣ صور، وغيرنا Visible لأن تكون Falseوأردنا عند اختيار رقـــم ١ Option1 عــرض الصـــورة الأولى وهكذا، نكتب الكود في Option1 كما يلي:

```
Private sub Option1.Click()
Image1.fisible = True
Image1.fisible = False
Image1.fisible =  False
End sub
```

والقيمة True تعنى أن الكائن محدد.

كرر العمل مع Option2 و Option3 مع تغيير ظهور الصورة.

صناديق الاختيار Check Box

لاختيار عدة اختيارات في نفس الوقت.

ويتم تنفيذ الأوامر المرتبطة بالصندوق عند اختياره.

فمثلا إذا كانت علامة صح بالصندوق، نرغب في عرض صورة معينة، نكتب الكود التالي:

```
Private Sub Command1_Click()
If Check1.value = 1 Then
Image1.Visible = True
End If
End Sub
```

وتكون قيمة الصندوق

- ٠ إذا كان لم يتم اختياره Unchecked.
- ١ إذا تم اختياره.
- ٢ إذا كان غير عامل مظلل Grayed.

أداة الصور PictureBox

تمكنك من وضع الصور داخل تطبيقاتك ومعالجة هذه الصور وتحريكها. واهم خصائص هذه الأداة:

Name	خاصية الأسم
Align	وضع الصورة على يمين الأطار ام على يساره
Appearance	طريقة الظهور
AutoRedraw	اعادة الرسم التلقائى
AutoSize	تحجيم الأطار تلقائيا بحجم الصورة الموضوعة فيه
BackColor	اللون الخلفى للأطار
BorderStyle	تحديد الشكل الخارجى للاطار
Enabled	هل الصورة فعالة أم لا
Height	خاصية ارتفاع الاطار
Left	خاصية بعد الاطار عن اقصى يسار النموذج

MouseIcon	شكل ايقونة الفارة من الاشكال التلقائية مثل الساعة الرملية سهمى تغيير العرض
MousePointer	اختيار شكل مؤشر الفارة من ايقونة خارجية
Picture	اختيار صورة ووضعها داخل اطار الأداة
ToolTipText	نص المساعدة الذى يظهر لو توقفت بالفرة أعلى الصورة لمدة قصيرة
Top	بعد الصورة من اعلى النموذج
Visible	الصورة ظاهرة ام مخفية
Width	عرض الصورة

تحميل الصور اثناء التشغيل:

هنك عدة تطبيقات تعتمد على صور خارجية، مثلا تطبيق متصفح الصور حيث تحمل الصور من ملفات خارجية ويمكن ذلك بأستخدام أمر LoadPicture بالصيغة التالية:

```
Picture1.picture = LoadPicture(Picture Path)
```

ويجب ان يكون المسار للصورة كاملا.

قلب الصورة

١- يمكنك نسخ صورة ولصقها مقلوبة. ضع عدد ٢ PictureBox واحد به الصورة والآخر للذى ستنقل اليه الصورة مقلوبة. واستخدم الكود التالى:

```
Private Sub Command1_Click( )
Picture2.PaintPicture Picture1.Picture,0,0,_
Picture1.Width,Picture1.Height,0,0,_
Picture1.Width,Picture1.Height, VbSrcCopy
End Sub
```

٢- الوضع الأفقى

```
Private Sub Command2_Click( )
Picture2.PaintPicture Picture1.Picture,0,0,_
Picture1.Width,Picture1.Height,Picture1.Width,_
0,-Picture1.Width,Picture1.Height, VbSrcCopy
End Sub
```

٣- الوضع الرأسى

```
Private Sub Command3_Click( )
Picture2.PaintPicture Picture1.Picture,0,0,_
Picture1.Width,Picture1.Height,0,Picture1Height,_
Picture1.Width, -Picture1.Height, VbSrcCopy
End Sub
```

٤- قلب الصورة

```
Private Sub Command4_Click( )
Picture2.PaintPicture Picture1.Picture,0,0,_
Picture1.Width,Picture1.Height,Picture1.Width,_
Picture1.Height, -Picture1.Height, -Picture1.Height,
VbSrcCopy
End Sub
```

أداة الصور IMAGE

أداة الصور هى ثانى اداة تمكنك من وضع الصور داخل تطبيقاتك، وتتميز عن اداة Picture فى بعض الخصائص. واهم خصائص هذه الأداة:

Name	خاصية الأسم
Appearance	طريقة الظهور
AutoSize	تحجيم الأطار تلقائيا بحجم الصورة الموضوعة فيه
BorderStyle	تحديد الشكل الخارجى للاطار
Enabled	هل الصورة فعالة أم لا
Height	خاصية ارتفاع الاطار
Left	خاصية بعد الاطار عن اقصى يسار النموذج
MouseIcon	شكل ايقونة الفارة من الاشكال التلقائية مثل الساعة الرملية سهمى تغيير العرض
MousePointer	اختيار شكل مؤشر الفارة من ايقونة خارجية
Picture	اختيار صورة ووضعها داخل اطار الأداة

Stretch	تحجيم الصورة بحجم الأداة
ToolTipText	نص المساعدة الذى يظهر لو توقفت بالفرة أعلى الصورة لمدة قصيرة
Top	بعد الصورة من اعلى النموذج
Visible	الصورة ظاهرة ام مخفية
Width	عرض الصورة

وتتميز هذه الأداة بخاصية Stretch والتى لا تتوفر فى الأداة PictureBox لذلك نفضل استخدام هذه الأداة. ويمكنك تحميل الصور اثناء التشغيل بأستخدام الأمر LoadPicture.

الرسائل Messages

من أهم الكائنات استخداما في فيجوال بيسك، والكود الخاص بالرسالة كما يلي:

```
Private Sub Command1_click()
Msgbox "Message",x,"Title"
End Sub
```

عند الضغط على الزرار ستظهر الرسالة ويمكن وضع الكود في تايمر.

Message هي الرسالة المطلوبة

X هي إما نوع الازرار أو الصورة بجانب الرسالة وتكون بالأرقام كما يلي:

Shapes:
- 16 Stop
- 32 Question Mark
- 48 !
- 64 Help

Buttons:
- 1 Ok // Cancel
- 2 Retry // Abort // Ignore
- 3 Cancel // Yes // No

4 Yes // No
5 Cancel // Retry

عند وضع الرقم X يجب أن يكون للصورة أو للازرار وعند اختيارها كصورة تكون الازرار Ok.

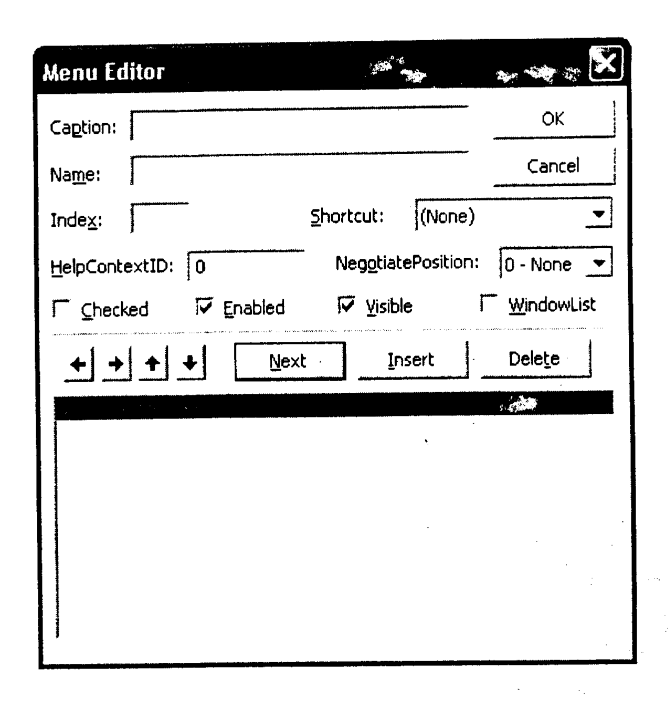
من المهم في الرسائل استخدام الأمر Response التعامل مع الازرار فى الرسـالة فإذا أردت وضع زرار خروج وعند الضغط عليه تظهر رسالة هل تريد الخـروج؟ وعند الضغط على Yes يخرج و No يبقى في البرنامج، اكتب الكود التالي:

```
Private Sub Command1_Click()
Response = MsgBox ("Are you sure?",VbYesNo,"Exit")
If Response = vbYes Then
End
ElseIf Response = vbNo Then
Form1.show
End If
End Sub
```

وهذا يعنى انه إذا كانت الاستجابة Yes فيخرج وإذا كانت No فتظهر النموذج ويبقى في البرنامج

استخدام القوائم Menu Editor

لإضافة قائمة إلى البرنامج اختار Menu Editorمن قائمة Tools في صـف القوائم سيظهر الصندوق التالي:



محتويات صندوق القوائم:

١- بجانب كلمة **Caption** يوضع ما يكتب في القائمة

٢- بجانب كلمة **Name** نكتب اسم القائمة

٣- بجانب **Shortcut** نقوم باختيار الاختصار مثل **Ctrl+N**

٤- **Checked** معناها أن التعامل بالقائمة سيكون بالعلامات

٥- **Enabled** وهي تعني ما إذا كانت القائمة متاحة أم لا

٦- **Visible** وتعني ما إذا كانت القائمة ظاهرة أم لا.

وإذا أردت أن تفتح قائمة من قائمة أخرى اضغط على السهم الأيمن بجانب **Next** ولفتح قائمة جديدة من القائمة السابقة أو للعودة إلى القائمة الرئيسية اضغط على السهم الأيسر بجوار **Next**.

وإذا أردت وضع خط تحت احد الحروف في الكلمة لاستخدامه مع **Alt** ضع & قبـــل الحرف المطلوب وضع خط تحته.

أعمدة التحرك Scroll Bars

يجب لاستخدام **Scroll Bars** في برنامجك أن تضع شيء ترغب في تحريكـــه مـــثلا لتحريك صورة يمينا ويسارا يجب وضع **HScroll1** وهو عمود التحريـــك الافقـــي وكتابة الكود التالي:

```
Private Sub HScroll1_change
Image1.Left = HScroll1.Value
End Sub
```

ولتحريكها لأعلى ولأسفل نضع **VScroll1** ونكتب الكود التالي:

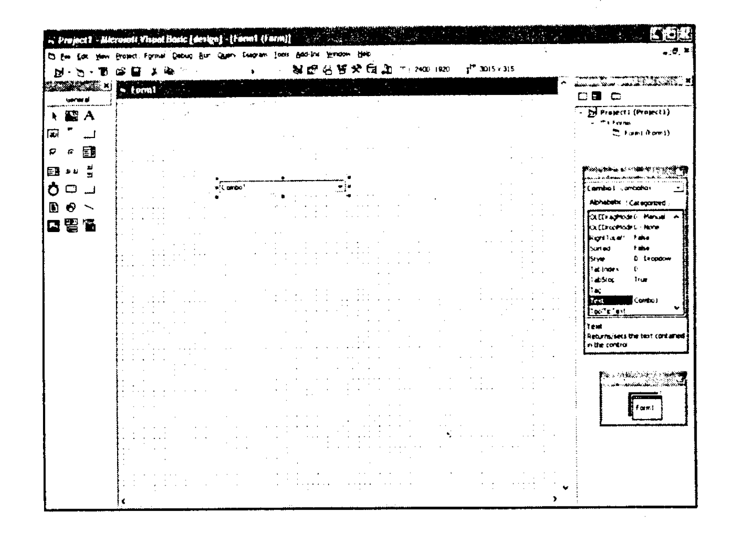
```
Private Sub VScroll1_change
Image1.Topeft = VScroll1.Value
End Sub
```

صندوق ألCombo Box & List Box

أولا: Combo Box

يكون شكل الأداة كما يلي عند وضعها على النموذج

ولإضافة أسماء للصندوق وكتابة الكود له ننفذ ما يلي:



١- لوضع اختيارات في الصندوق نكتب ما يلي:

```
Private Sub Form1_Load()
Combo1.AddItem " Mohamed"
Combo1.AddItem " El Fayuomi"
End Sub
```

وهذا الكود يتزود الصندوق بكلمتين Mohamed و El Fayuomi ويمكنـك إضافة أسماء أخرى لنفس الكود.

٢- لكتابة الكود عليك تغيير Change الموجودة بصفحة الكود إلى Click ثم اكتب الكود التالي

```
Private Sub Combo1_Click()
Select case Combo1.ListIndex
Case 0
Expression
Case 1
Expression
```

```
End Select
End Sub
```

استخدمنا قاعدة Select Case وبدلنا الحالات ب Case 0 وليس Case1.

Case 0 مسئولة عن الاسم Mohamed أما Case1 فمسئولة عن الاسم El Fayuomi.

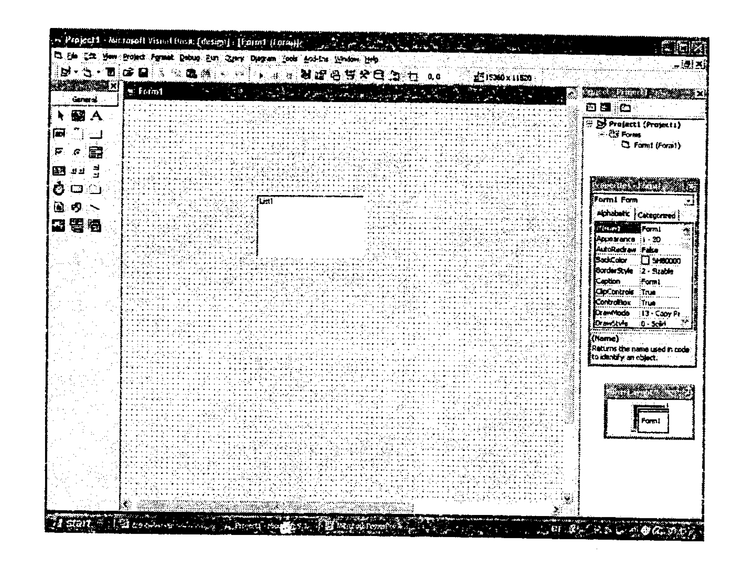
Combo1.ListIndex تعنى انه سيتم التحكم في عناصر الصندوق.

وكلمة Expression تعنى انه يمكنك كتابة اى جمل مثلا عرض الصورة الخاصة بمحمد Image1.Visible = True أو تغيير عنوان مثل Label1.Caption = Mohamed.

نهاية الكود End Select تنهى الاختيارات.

ثانيا: List Box

يكون شكل الأداة كما يلي عند وضعها على النموذج، ولإضافة أسماء للقائمة وكتابة الكود له ننفذ ما يلي:



١- لوضع اختيارات في هذا الصندوق نكتب الكود التالي:

```
Private Sub Form1_Load()
List1.AddItem " Mohamed"
List1.AddItem " El Fayuomi"
End Sub
```

وبهذا الكود تتزود القائمة بكلمتين **Mohamed** و **El Fayuomi** ويمكنك إضافة أسماء أخرى لنفس الكود.

٢- لكتابة الكود عليك تغيير **Change** الموجودة بصفحة الكود إلى **Click** ثم اكتب الكود التالي:

```
Private Sub List1_Click()
Select case List1.ListIndex
Case 0
Expression
Case 1
Expression
End Select
End Sub
```

استخدمنا قاعدة **Select Case** وبدئنا الحالات ب **Case 0** وليس **Case1**.

Case 0 مسئولة عن الاسم **Mohamed** أما **Case1** فمسئولة عن الاسم **El Fayuomi**.

List1.ListIndex تعنى انه سيتم التحكم في عناصر الصندوق.

وكلمة **Expression** تعنى انه يمكنك كتابة اى جمل مثلا عرض الصورة الخاصة بمحمد **Image1.Visible = True** أو تغيير عنوان مثل **Label1.Caption = Mohamed**.

نهاية الكود **End Select** تنهى قاعدة **Select** الاختيارات.

فتح برنامج

يمكن فتح برنامج عن طريق استخدام فيجوال بيسك لأمر **Shell** كما يلي:

```
Private Sub Command1_Click
```

```
A= shell("C:\windows\notepad.exe",vbNormalFocus)
End Sub
```

وعند الضغط على الزرار سوف يفتح برنامج **notepad** ويلاحظ اســتخدام المســار الصحيح للبرنامج المطلوب

مسح الملفات

لمسح اى ملف في مكان معين نستخدم أمر **Call** كما يلي:

```
Private Sub Command1_Click
Call Kill ("Path")
End Sub
```

وتكون **Path** هي مسار الملف مع كتابة اسمه

```
Private Sub Command1_Click
Call Kill("c:\document.txt")
End Sub
```

ولمسح كل الملفات ذات الامتداد المعين نستخدم * مثلا **Wav.*** لمسح كل ملفــات **wav** من المسار المحدد كما يمكن استخدام *.* لمحو كل الملفات ويلاحظ أنها ســتمحى نهائيا.

خصائص أخرى للملفات

١- إخفاء الملفات

```
Private Sub Command1_Click
SetAttr ("Path"), vbHidden
End Sub
```

٢- جعل الملفات للقراءة فقط **Read Only**

```
Private Sub Command1_Click
SetAttr ("Path"), vbReadOnly
End Sub
```

٣- حفظ الملفات كأرشيف

```
Private Sub Command1_Click
SetAttr ("Path"), vbArchive
End Sub
```

ونكتب مكان ("Path") مسار الملف واسمه.

التصادم بجدران النموذج

لتحريك الصورة أو الكائن أفقيا إلى أن يصطدم بجدار الشاشة فيعود ثانية نستخدم المثال التالي:

انشيء مشروع به نموذج وتايمر وصورة.

خصائص النموذج:

Name =FrmMain

خصائص التايمر:

Name = TmrMove

Interval = 100

خصائص الصورة:

Name = PicMove

يجب معرفة أقصى نقطة تتحرك فيها الصورة إلى اليسار، وأقصى نقطة تتحرك فيها الصورة إلى اليمين.

أقصى نقطة تتحرك فيها الصورة إلى اليسار هي النقطة صفر حيث تكون الصورة ملاصقة لأقصى يمين النموذج.

وأقصى نقطة تتحرك فيها الصورة إلى اليمين فتحتاج إلى معرفة:

١- أقصى نقطة علي يمين النموذج هي **Form.Width** اى عرض النموذج.

٢- أقصى نقطة علي يمين الصورة هي **Picture.Width** مضافا إليها **Picture.left**

٣- أقصى نقطة تتحرك عندها الصورة إلى اليمين هي التي يكون عندها **Picture.left + Picture.Width** أقل من أو يساوى **Form.width**.

نعلن عن متغير يمثل سرعة الصورة لليمين ولليسار بحيث لا يزيد عن ١٠٠ وسنستخدم النوع **Integer** حيث سيحمل قيم سالبة في بعض الحالات.

في إجراء تحميل النموذج سنجعل مكان الصورة إلى أقصى يسار النموذج وذلك لأن الحركة ستبدأ من اليسار إلى اليمين.

```
Picmov.left= 0
```

تحديد نموذج البدء Startup

- عند إضافة المبرمج لأكثر من نموذج **Form** وعندما يقوم بالتنفيذ **Run** لن يعمل إلا **Form1**. ولحل هذه المشكلة نقوم بالخطوات التالية:

١- نفتح قائمة **Project** ---> **Project Properties** ---> تظهر نافذة **Project Properties**.

٢- أمام عبارة **Startup Object** نختار النموذج و ليكن **Form1** من الصندوق الاختيارات **Combo box** التي سوف يتم تشغيلها عند التنفيذ.

- أما بالنسبة للمستخدم يتم حل هذه المشكلة عن طريق إنشاء القوائم المنسدلة أو ازرار الأوامر **Command Buttons**.

مثال:

١- بفرض أن لدينا نموذجان Forn1 و Form2 .

٢- ننشط Form1 و نجعله Startup ثم نضع عليه زرار أمر Command Button نكتب على هذا الزرار " فتح شاشة أخرى ".

٣- Double click على زرار الأمر ونكتب الكود الأتي:

```
Form2.show
Form1.hide
```

أمر Rem

هو أمر يستخدمه المبرمج في حالتين:

١. لإبطال مفعول أمر أو عدة أوامر نشك في عدم صحتها و في هذه الحالة سوف يتم يتجاهلها بواسطة فيجوال بيسك.

٢. كتابة تعليقات أو ملاحظات للمبرمج.

صيغة الأمر Rem

١- قبل الأمر نكتب Rem. معني هذا أن هذه جملة ليست للاستخدام، ويمكن كتابة ' بدلاً من الأمر Rem ليقوم بنفس العمل.

التفريع والدورات

قاعدة If

هي قاعدة تساعد في اتخاذ قرار وبناءا على إجراء معين، فإذا تحقق الشرط يتخذ إجراء معين، وإذا لم يتحقق الشرط يتخذ إجراء من نوع آخر.

قاعدة If البسيطة:

تتكون هذه القاعدة مما يلي:

١- If: وهى أول شيء يكتب في الكود ومعناها انه إذا تحقق الشرط المعين.

٢- Then: تكتب بعد الشرط الموجود بعد If ومعناها إذا تحقق الشرط الموجود بعد If فيتم تنفيذ ما بعد Then.

٣- **Else** : إلا إذا لم يتحقق الشرط الموجود بعد **If** يتم تنفيذ ما بعد **Else** وهى ليست إجبارية حيث يمكن عدم استخدامها في بعض الحالات.

٤- **ElseIf**: تستخدم لوضع شرط جديد فهي مثل **If** ولكن بدلا من أن نكتب كود جديد به **If** و**End If** نكتب **ElseIf** ونستكمل الكود.

٥- **End If** : تكتب في نهاية الكود لإنهاء القاعدة وهى إلزامية

تطبيق:

```
Private Sub Command1_Click( )
If Text1.Text= "123" Then
Image1.Visible = True
ElseIf Text1.Text= "    " Then
Image2.Visible = True
Else
Image2.Visible = True
End If
End Sub
```

يتضح من الكود انه يماثل استخدام كلمة السر **Password** عند كتابتها صحيحة تظهر صورة محددة وعند كتابتها خطأ تظهر صورة أخرى.
واستخدمت كلمة **123** ككلمة سر إذا تم كتابتها في **Text1.Text** ينفذ الحاسب ما بعد **Then** أي يعرض **Image1** وإذا تم إدخال مسافات أو اى حروف أخرى يعرض الصورة **Image2**.

وتستخدم إذا كان هناك احتمالين فقط مثل نتيجة الطالب إما " ناجح أو راسب ".

الصيغة:

IF الشرط then

ملاحظات:

١- استخدمنا الدالة **Val** اختصارا لـ **Value** والتي تحول الحرفي إلى رقمي.

٢- استخدمنا الدالة **Str** اختصارا لـ **String**، لتحويل القيم من نوع رقمي **Integer** إلى النوع **String** حيث لن ينفع وضع دالة **N** من نوع حرفي مع دالة **B** و هي من نوع رقمي.

٣- لابد من كتابة الكود داخل المكان الذي سوف ينطلق منه الحدث أي زرار الأمر.

:Nested IF

تستخدم إذا تعددت الاحتمالات مثل (شرائح العمولة - تقديرات الطلبة - شرائح الضرائب)

الصيغة العامة للقاعدة:

IF الشرط **then**

أمر أو عدة أوامر تنفذ إذا تحقق الشرط

Elseif الشرط **then**

أمر أو عدة أوامر تنفذ إذا تحقق الشرط

Elseif الشرط **then**

أمر أو عدة أوامر تنفذ إذا تحقق الشرط

Else ثم في النهاية

أمر أخر إذا كان ضرورياً

End IF

مثال: "برنامج لتقديرات الطلبة" علماً بان:

اقل من ٥٠ (راسب).

من ٥٠ - ٦٠ (مقبول).

أمر أو أوامر إذا تحقق الشرط

أي غير ذلك Else

أمر أو أوامر إذا لم يتحقق الشرط

End IF

برنامج على IF البسيطة:

مطلوب عمل الأتي:

١- إضافة نموذج جديد و جعله Startup.

٢- نضع على النموذج ---> Text box و أمامه Label1 مكتوب عليه " ادخل اسم الطالب ".

٣- نضع على النموذج ---> Text box و أمامه Label2 مكتوب عليه " درجة الطالب ".

٤- نضع على النموذج ---> Label2 مكتوب عليه " النتيجة ".

٥- نضع على النموذج ---> Command Button مكتـــوب عليـــه " احصل على النتيجة ".

بفرض أن شرط النجاح هو حصول الطالب على ٦٠ درجة أو أكثر. فإذا تحقق الشرط نكتب عبارة " ناجح "، و إذا لم يتحقق الشرط نكتب عبارة " راسب ".

يكتب الكود التالي بعد الضغط على زرار الأوامر Command Button

```
Dim N as String, B as Integer< R as String
N = Text1
B = Text2
IF Val(B) >= 60 then
R = " ناجح "
Else
R = " راسب "
End IF
Label3.caption = N + " " + R + " " + " و قد حصل على " +
Str(B)
```

أكبر من ٦٠ – ٧٥ (جيد).

أكبر من ٧٥ – ٨٥ (جيد جدا).

أكبر من ٨٥ – ١٠٠ (امتياز).

الكود:

```
Private Sub Command1_Click( )
Dim n As String, d As Integer, r As String
n = Text1
d = Text2
If Val(d) < 50 Then
r = "راسب"
Elseif Val(d) <= 60 Then
r = "مقبول"
Elseif Val(d) <= 75 Then
r = "جيد"
Elseif Val(d) <= 85 Then
r = "جيد جيدا"
Elseif Val(d) <= 100 Then
r = "امتياز"
End If
Label3.Caption = n + " " ÷ r + " " + "بمجموع درجات" + " "
+ Str(d)
End Sub
```

أمر Select Case

هو أمر بديل لقاعدة IF بشكليها البسيطة و المتداخلة.

صيغة Select Case البسيطة:

Select Case المتغير الذي نريد اختباره

Case is شرط

أمر أو عدة أوامر تنفذ إذا تحقق الشرط

Case Else

أمر أو عدة أوامر إذا لم يتحقق الشرط

End Select

مثال

برنامج العمولة كتطبيق لقاعدة Select Case البسيطة

الشرط هو أن تحقق مبيعات ٦٠٠٠ ج أو أكثر:

- إذا تحقق الشرط تكون هناك عمولة ٦%.
- إذا لم يتحقق الشرط تكون العمولة ١%.

مرحلة التصميم:

- مطلوب إضافة نموذج جديد إلى نفس المشروع ثم جعله Startup Object ثم نحفظ النموذج باسم " العمولة ".
- Textbox1 و نضع أمامه Label1 مكتوب عليه " اسم البائع ".
- Textbox2 و نضع أمامه Label2 مكتوب عليه " قيمة المبيعات ".
- Label3 مكتوب عليه " العمولة ".
- Command Button مكتوب عليه " عمولة البائع ".

الكود:

```
Private Sub Command1_Click()
Dim n As String, s As Integer, r As Integer
 n = Text1
 s = Text2
 Select Case Val(s)
 Case Is >= 6000
 r = Val(s) * 0.06
```

```
Case Else
r = Val(s) * 0.01
End Select
Label3.Caption = n + " " + "حقق عمولة" + Str(r) + " " +
"و قد حقق مبيعات قدرها" + " " + Str(s)
End Sub
```

Select case المتداخلة:

Select Case المتغير الذي نريد اختباره

Case is شرط

أمر أو عدة أوامر إذا تحقق الشرط

Case is شرط

أمر أو عدة أوامر إذا تحقق الشرط

و هكذا،،،،،

End Select

مثال: برنامج تقديرات الطلبة

الكود:

```
Private Sub Command1_Click()
Dim n As String, b As Integer, r As String
n = Text1
b = Text2
Select Case Val(b)
Case Is < 50
r = "راسب"
Case Is <= 60
r = "مقبول"
Case Is <= 75
r = "جيد "
Case Is <= 85
r = "جيد جدا"
Case Is <= 100
```

```
r = "امتياز"
End Select
Label3.Caption = n + " " + r + " " + "و قد حصل على" +
Str(b)
End Sub
```

تمارين:

١. عرض نتيجة الطالب ناجح أو راسب باستخدام **If** و **Select Case**.
٢. تنفيذ برنامج العمولة باستخدام **If** البسيطة و**Select Case**.
٣. تصميم برنامج تقديرات الطلبة باستخدام **If** المتداخلة و **Select Case**.

التكرار باستخدام أمر Do ...... Loop Until

تستخدم قاعدة **Do .... Loop Until** لتكرار مجموعة أوامر إلى أن يحدث حدث ما، وتتكون من:

١. **Do**: وهى أول الأمر.
٢. **Loop Until**: تكتب بعد الجمل التي نرغب في تكرارها، ومعناها أن هذه الجمل سيتم تكرارها إلى أن يحدث ما بعد**Until**.

مثال على الكود:

```
Private Sub Command1_Click( )
Score.Caption = "0"
Do
Score.Caption = Score.Caption +1
Loop Untill Score.Captio = "10"
End Sub
```

جرب وضع هذا الكود أيضا في **Timer** وعدل **Interval** إلى اى رقم تريده (الثانية = ١٠٠٠)

أغلاق البرنامج

سنكتب برنامج يحتوى على ما يلي:

- كائنين: كائن نموذج وكائن زرار أمر.
- حدث: حدث الضغط على زرار أمر.
- إجراء: إغلاق البرنامج.

نفذ الخطوات التالية:

- أفتح مشروع جديد.
- أضف زرار أمر للنموذج.
- غير اسم زرار الأمر إلى **CmdEnd** .
- غير خاصية **Caption** للزرار لتصبح **Exit** .
- أضغط على زرار **F5** ولاحظ عدم حدوث شيء لأننا لم نكتب الكود الخـــاص بالزرار.
- أرجع إلى البرنامج واضغط ضغطتين على الزرار، واكتب الكود التالي به:

```
Private Sub CmdEnd_click( )
End
End Sub
```

نفذ البرنامج واضغط على زرار الأمر يتم الخروج من البرنامج.

أمر While Wend

تستخدم هذه الدورة لتنفيذ شيء معين في حالة حدوث حدث معين، وعند انتهاء هذا الشيء ينتهي الحدث، وتتكون القاعدة من **While, Wend**

مثال على كتابة الكود:

```
Private Sub Command1_klick()
```

```
Score.Caption = "0"
While  Score.Caption < 10
Score.Caption = Score.Caption +1
Wend
End Sub
```

إذا لم يعمل هذا الكود ضعه في Timer وحدد الفاصل الزمني Intervals بجزء من الثانية.

تحديد المسارات Path

يقصد بالمسارات استخدام كل من DriveListBox و DirListBox و FileListBox وتستخدم بدلا من شاشة فتح الملف عن طريق CommonDialog ولربط البيانات مع بعضها نكتب الكود التالي في كل من:

```
Drive1:
Dir1.Path = Drive1.Drive
Dir1:
File1.Path = Dir1.Path
File1:
SelectedFile = File1.Path & "\" & File1.filename
LoadedFile
```

في الكود الخاص FileListBox عند كلمة LoadedFile أي انه نوع الملف الذي سيفتح سواء كان bmp أو txt .

في خاصية File1 ستجد خاصية باسم Pattern ومكتوب بجانبها *.* عليك تغييرها حسب نوع الملف الذي يفتح فإذا كانت صورة نغيرها إلى bmp.* وإذا كانت txt نغيرها إلى txt.* .

الفصل التاسع

بعض البرامج التطبيقية

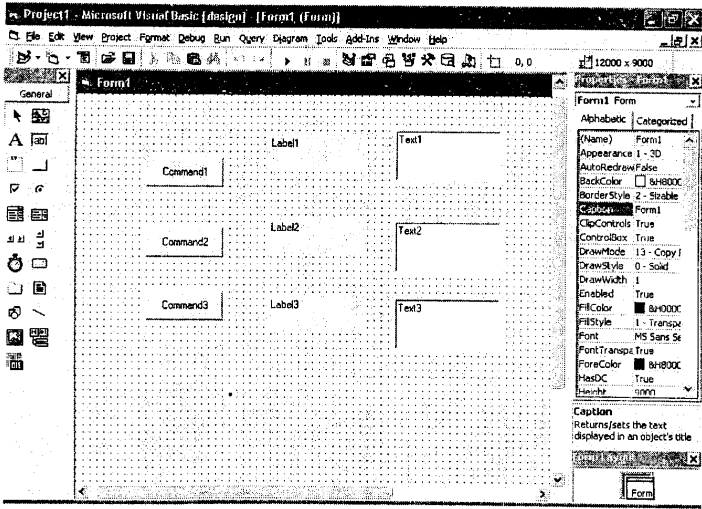
برنامج الميقاتي

برنامج الميقاتي يستخدم الساعة الداخلية للحاسب، ويهدف إلى معرفة الفارق الزمني بين بداية التوقيت وانتهاؤه، وسنتعرف علي كيفية التحكم بالأزرار.

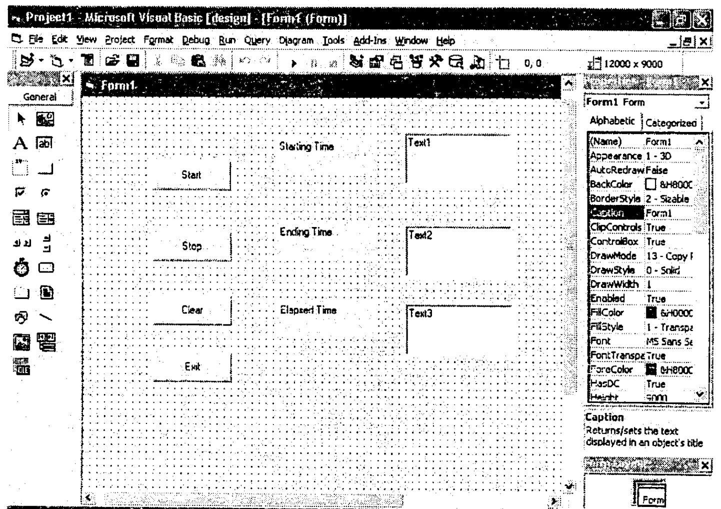
احتياجات البرنامج:

تعتمد البرمجة بلغة فيجوال بيسك علي وضع أشكال **Objects** بشاشة الحوار **Form** ثم تعديل هذه الأشكال وكتابة الأوامر الخاصة بكل شكل، والأشكال التي تحتاجها إليها لتصميم هذا البرنامج هي:

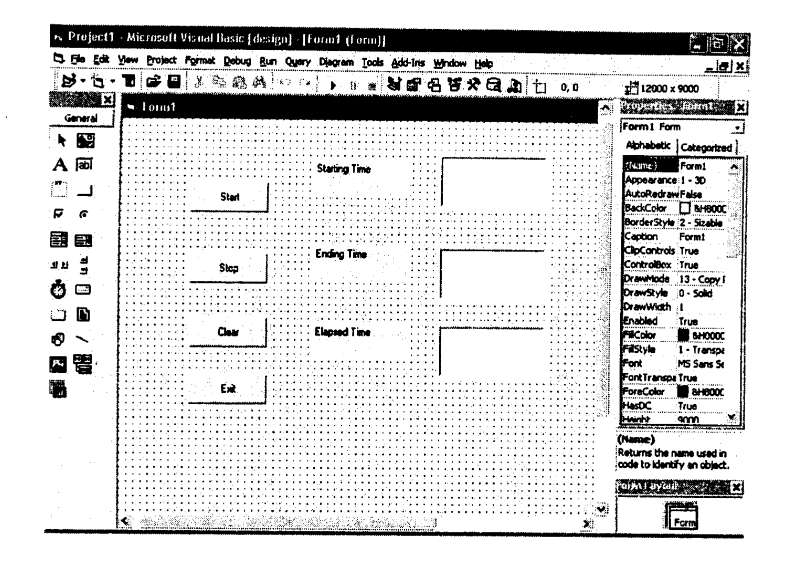
- أزرار الأوامر **Command Buttons** عدد ٢.
- صناديق كتابة العناوين **Labels** عدد ٣.
- صناديق استقبال / عرض البيانات **Text** عدد ٣.



يعد وضع الأشكال عدل خصائص جميع الأشكال التي أضيفت إلى شاشة الحوار بوضع الفارة بكل شكل ثم الضغط علي مفتاح **F4**، فتظهر لك شاشة تغيير الخصائص، فيتم تعديل العنوان باختيار **Caption** لتعديل اسم المشروع عند استدعائه.



كذلك بالنسبة لأزرار الأوامر **Command Buttons** وصناديق العناوين **Labels** أما صناديق البيانات **Text** فيتم التعديل باختيار أمر **Text**. بعد التعديل يجب أن يكون شكل شاشة النموذج كما يلي:



تعديل أسماء الأشكال:

كل شكل له اسم معين مثل **text1** و **Text2** وهذه الأسماء افتراضية من البرنـــامج، ومن المفيد إعادة تسمية هذه الأشكال لنستطيع التعرف عليها أثناء كتابة أوامر البرنامج، وتتم التسمية كما يلي:

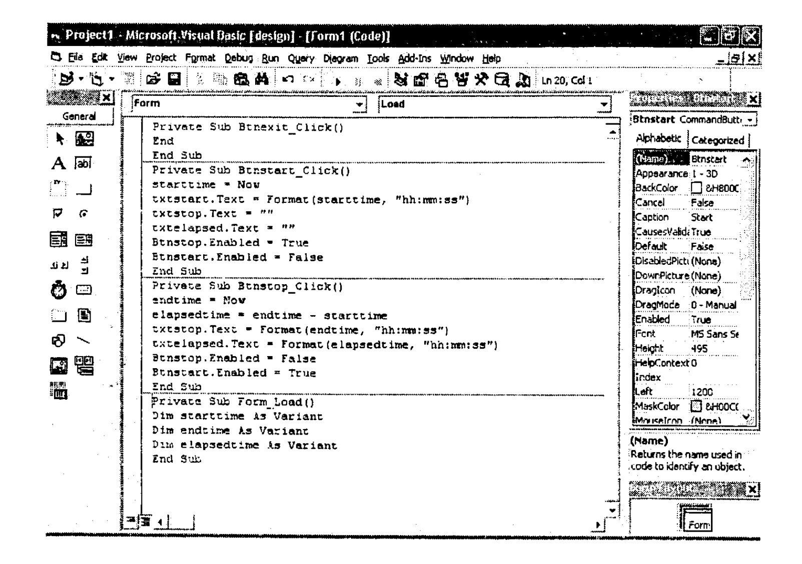
١- تحديد الشكل المطلوب.

٢- اختيار صفة **Name** من شاشة الخصائص **Properties**

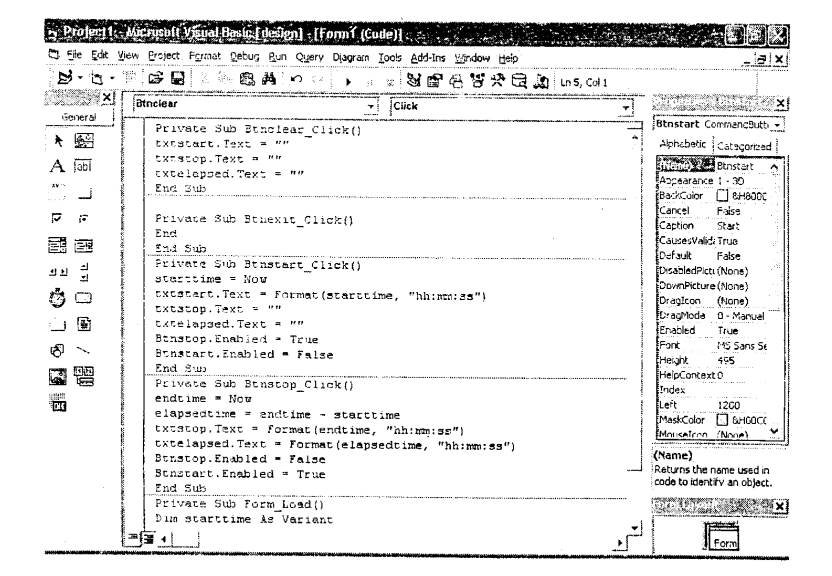
٣- كتابة الاسم الجديد مكان الاسم السابق، ثم اضغط علي مفتاح إدخال.

غير أسماء الأشكال التالية:

- اسم الشكل **Command1** إلي **Btnstart**.
- اسم الشكل **Command2** إلي **Btnstop**.



اسم الشكل Text1 إلى **Txtstart**.



اسم الشكل Text2 إلى **Txtstop**.

اسم الشكل **text3** إلى **Txtelapsed**.

ملحوظة:

تستطيع معرفة اسم الشكل السابق باختيار الشكل، وسنجد بشاشة الخصائص أن الاسم يظهر في السطر الأول من هذه الشاشة.

كتابة أوامر البرنامج:

بعد إعداد الأشكال بشاشة الحوار **Form** وتعديل الخصائص الخاصة بالإشكال تتم كتابة أوامر البرنامج، نفذ الخطوات التالية:

١- ضع الفارة علي زرار **start**.

٢- اضغط زرار الفارة مرتين لفتح شاشة كتابة أوامر الشكل.

٣- اكتب الأوامر الخاصة بالشكل كما يلي:

٤- لكتابة الأوامر الخاصة بالشكل التالي والمعنون **Stop** اختار أسماء الأشكال من الشاشة التي تستخدم لكتابة الأوامر، حيث ستجد عنوان **Object** والـــذي يشير إلى أسماء الأشكال التي وضعتها بشاشة الحوار، حدد اسم الشكل إلهاء ثم اكتب الأوامر الخاصة به كما يلي:

ثم اختار اسم الشكل **Form** واكتب الأوامر الخاصة به كما يلي:

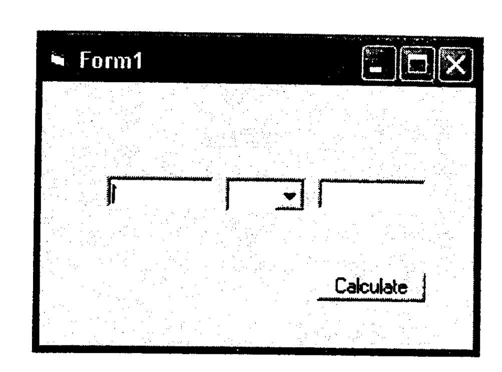
```
Sub btnstart_Click ( )
        Starttime = Now
        Txtstart.Text = Format(starttime, "hh: mm: ss")
        Txtstop.Text = "   "
        Txtelapsed.Text = "   "
        Btnstop.Enabled = True
        Btnstart.Enabled = False
End Sub

        Sub btnstop_Click ( )
        Endtime = Now
        Elapsedtime = endtime – starttime
        Txtstop.Text = Format (elapsedtime, "hh: mm: ss")
                Btnstop.Enabled = False
                Btnstart.Enabled = True
End Sub

        Sub Form_Load ( )
        Dim statime As Variant
        Dim endtime As Variant
        Dim elapsedtime As Variant
End Sub
```

برنامج للعمليات الحسابية الأربعة باستخدام كومبو بوكس

The Four Basic Math operations with combo box to select from




```
Option Explicit

Private Sub Form_Load()
   cmbOperator.AddItem ("+")
   cmbOperator.AddItem ("-")
   cmbOperator.AddItem ("*")
   cmbOperator.AddItem ("/")
End Sub

Private Sub cmdCalculate_Click()
   If cmbOperator.Text = "" Then
      MsgBox ("You most chose a operator!!")
   Else
      Select Case cmbOperator.Text
         Case "+"
            lblResult.Caption      =      Val(txtNumber1.Text)      +
Val(txtNumber2.Text)
         Case "-"
         lblResult.Caption      =      Val(txtNumber1.Text)      -
Val(txtNumber2.Text)
         Case "*"
```

```
        lblResult.Caption      =      Val(txtNumber1.Text)      *
Val(txtNumber2.Text)
        Case "/"
        lblResult.Caption      =      Val(txtNumber1.Text)      /
Val(txtNumber2.Text)
      End Select
   End If
End Sub
```

برنامج لحساب اقسط اهلاك الآلات:

قسط الأهلاك = (تكلفة الأصل – القيمة كخردة) / سنوات حياته الانتاجية

مجمع الاهلاك = مجمع الاهلاك فى اول المدة + اهلاك العام

```
Private Sub Command1_Click()
Text5 = (Val(Text1) - Val(Text2)) / Val(Text3)
Text6 = Val(Text4) + Val(Text5)

End Sub

Private Sub Command2_Click()
Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""
Text5.Text = ""
Text6.Text = ""
End Sub

Private Sub Command3_Click()
For i = 1 To 10000
Print Tab(i), "goodbay"
Next i
End
End Sub
```

بعد كتابة البرنامج نفذه بالضغط علي مفتاح **F5** أو بالضغط علي زرار التشغيل الـــذي يظهر في اعلي الشاشة.

شرح أوامر البرنامج:

ينقسم البرنامج إلى ٣ أقسام وهي:

الإجراء **Sub Btnstart**: يحتوى هذا الإجراء على الأوامر المطلوب تنفيذها عنـــد الضغط علي زرار بدء.

الإجراء **Sub Btnstop**: يحتوى علي الأوامر المطلوب تنفيذها عند الضغط علي زرار انتهاء.

الشكل العام المعنون **Form**: يحوى هذا الجزء المتغيرات التي ستستخدم بـــالإجراءات المتعددة، يجب أن يتم تعريف الأشكال بهذا الجزء.

من الأوامر الجديدة في هذا البرنامج:

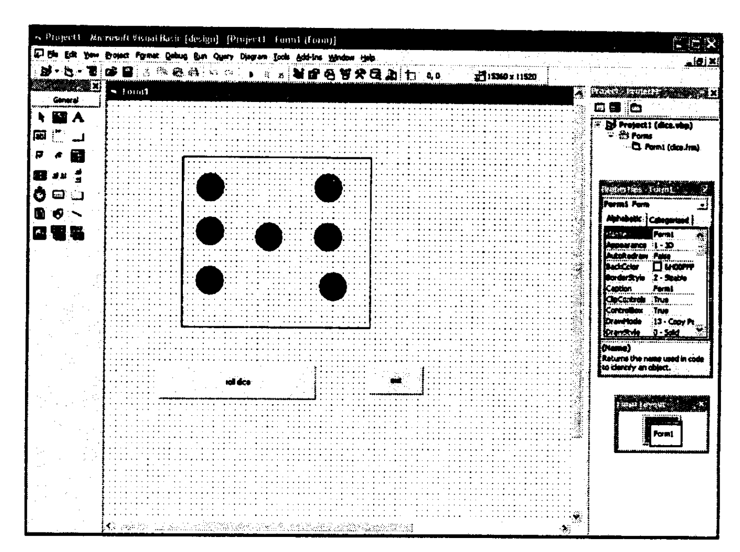
Starttime = Now وظيفة هذا الأمر هي قراءة الوقت الداخلي للنظـــام **Now** وتخزين الرقم بالموقع **StartTime**.

Format وظيفته استخراج الوقت فقط من الموقع **StartTime**، ذلك أن وظيفـــة **Now** هي نقل الوقت والتاريخ إلى الموقع **StartTime**، لهذا يجب أن نحدد بـــالأمر **Format** شكل البيانات المطلوب عرضها بالموقع الذي نختاره.

Btnstart.Enabled = False : وظيفة هذا الأمر تعطيل زرار بدء العمل مؤقتاً لحين إصدار أمر آخر **(True)** ، علي المبرمج تحديد المفاتيح المطلوب استخدامها أثنـــاء إنجاز عملية معينة أما لمفاتيح الاخري فتكون معطلة ولا يستطيع المستخدم اســـتعمالها إلا بعد إعطاء الأمر الذي يعيد تشغيلها.

برنامج على محاكاة القاء زهر به ٧ دوائر تظهر عشوائيا

Roll Dice
Using Shape tool, rnd, visible, and val
Use 7 circle shapes inside rectangle, and two command buttons



```
Private Sub Command1_Click()
Randomize Timer
n = Int(1 + Rnd * 6)
For i = 0 To 6
Shape2(i).Visible = False
Next

If n = 1 Then
Shape2(3).Visible = True
End If
If n = 2 Then
Shape2(2).Visible = True
Shape2(4).Visible = True
End If

If n = 3 Then
```

```
Shape2(2).Visible = True
Shape2(3).Visible = True
Shape2(4).Visible = True
End If

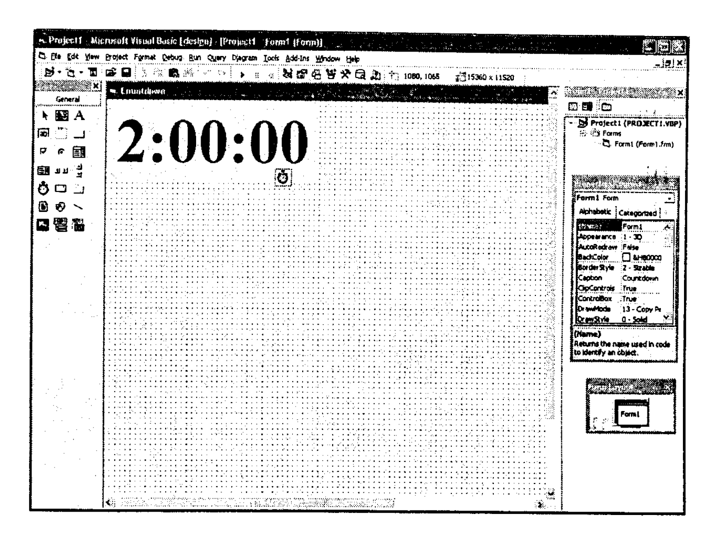
If n = 4 Then
Shape2(2).Visible = True
Shape2(0).Visible = True
Shape2(4).Visible = True
Shape2(6).Visible = True
End If

If n = 5 Then
Shape2(0).Visible = True
Shape2(2).Visible = True
Shape2(3).Visible = True
Shape2(4).Visible = True
Shape2(5).Visible = True
End If

If n = 1 Then
Shape2(0).Visible = True
Shape2(1).Visible = True
Shape2(2).Visible = True
Shape2(5).Visible = True
Shape2(4).Visible = True
Shape2(6).Visible = True
End If

End Sub

Private Sub Command2_Click()
End
End Sub
```




```
Option Explicit

Private AlarmTime As Date

Private Sub Form_Load()
   AlarmTime = DateAdd("h", 2, Now)
End Sub


Private Sub Timer1_Timer()
Dim txt As String

   txt = Format$(AlarmTime - Now, "h:mm:ss")
   Label1.Caption = txt
End Sub
```

برنامج لتصميم رسم حافظ للشاشة يتكون من ٣ دوائر ويتحكم فيها ٣ ميقاتي

```
Option Explicit

Dim i As Integer
Dim j As Integer
Dim a As Integer
Dim b As Integer
Dim r As Integer
Dim o As Integer
Private Sub Form_Click()
End
End Sub

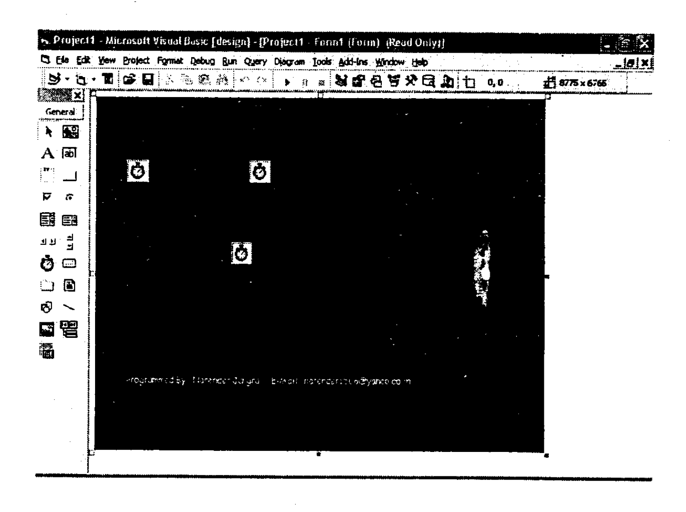
Private Sub Form_KeyPress(KeyAscii As Integer)
End
End Sub


Private Sub Form_Load()
Form1.BackColor = vbBlack
Form1.WindowState = 2
Label1.Top = 8500
Label1.Left = 11000
r = 300
End Sub


Private Sub Timer1_Timer()
With Label1
    If .Left <= 11000 Then .Left = .Left - 100
    If .Left < -5000 Then .Left = 11000
End With
End Sub

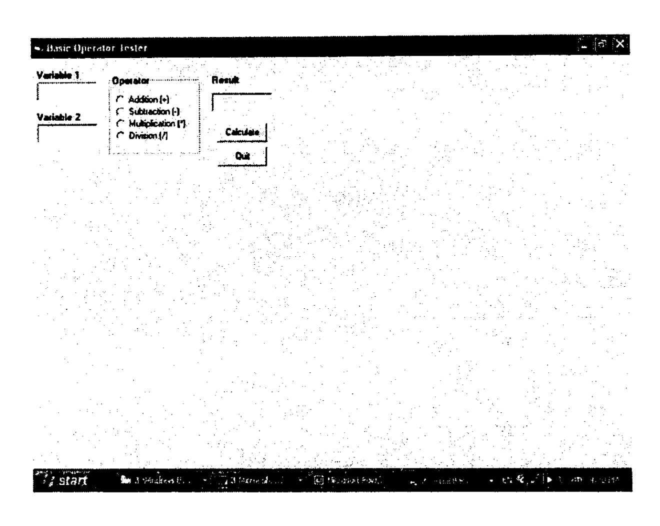
Private Sub Timer2_Timer()
Form1.BackColor = RGB(Rnd * 255, Rnd * 255, Rnd * 255)
End Sub
```

```
Private Sub Timer3_Timer()
Me.FillColor = RGB(Rnd * 255, Rnd * 255, Rnd * 255)
For i = 3000 To 10000 Step 20
Form1.Circle (i, 900), r
Form1.Circle (13000 - i, 900), r
Form1.Circle (10000, 10920 - i), r
Form1.Circle (3000, 10920 - i), r
Form1.Circle (10000, i - 2080), r
Form1.Circle (i, 7940), r
Form1.Circle (13000 - i, 7940), r
Form1.Circle (13000 - i, 10920 - i), r
Form1.Circle (i, i - 2080), r
Form1.Circle (i, 10940 - i), r
Form1.Circle (13000 - i, i - 2060), r
Form1.Circle (13000 - i, i - 2060), r
Form1.Circle (3000, i - 2080), r
Next i
End Sub
```



برنامج لحساب العمليات الحسابية:

استخدم ٢ تكست بوكس، فريم، ٤ شيك بتون، ٣ ليبل، ٢ كوماند بتون



```
Private Sub Command1_Click()
  Dim First, Second      'declare variables

  First = Val(Text1.Text) 'fetch and convert numbers
  Second = Val(Text2.Text)

  'if first button clicked, add numbers
  If Option1.Value = True Then
    Label1.Caption = First + Second
  End If
  'if second button clicked, subtract numbers
  If Option2.Value = True Then
    Label1.Caption = First - Second
  End If
  'if third button clicked, multiply numbers
  If Option3.Value = True Then
    Label1.Caption = First * Second
  End If
  'if fourth button clicked, divide numbers
  If Option4.Value = True Then
    Label1.Caption = First / Second
```

```
  End If
End Sub

Private Sub Command2_Click()
  End
End Sub
```

تكوين القوائم
وتصميم برنامج لمعالجة النصوص

للقوائم دوراً كبيراً بين المستخدم والحاسب، وتعمل جميع البرامج بالقوائم بطريقة أو أخري، مثلاً برامج ويندوز تعمل من خلال الأيقونات أو القوائم، فاستخدام القوائم بالبرامج يمكن المستخدم من تنفيذ البرامج التي تكتب بلغة فيجوال بيسك بطريقة أسرع وأسهل.

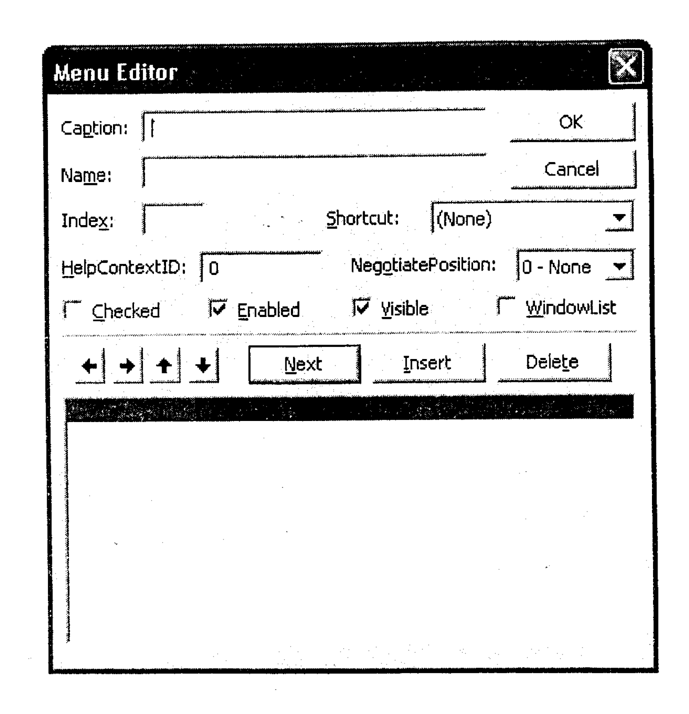
سندرس فيما يلي برنامج لتعلم كيفية تكوين القوائم والاختيارات وكتابة الأوامر المناسبة عند اختيار احد الاختيارات من هذه القوائم، وتعتمد فكرة البرنامج علي كتابة نص علي الشاشة، ثم باستخدام القوائم يتم تغيير حجم كتابة النص، أو تغيير أسلوب الكتابة أو لون الخط أي برنامج مصغر لمعالجة النصوص.

عناصر شاشة تكوين القوائم:

تساعدك الشاشة علي تكوين القوائم بطريقة سهلة ومباشرة، وإذا سبق لك واستخدمت برامج مثل **dbase** أو **Foxpro** فستدرك مدى الفارق في تكوين القوائم، حيث تبسط لغة فيجوال بيسك تكوين القوائم بطريقة لم تكن متاحة مسبقاً، فالتعديل والتكوين لا يستغرق سوي دقائق معدودة، ولم تعد كتابة البرامج مشكلة، فالمشكلة أساساً في تصميم البرنامج وليست في كتابته.

تكوين القوائم بلغة فيجوال بيسك:

١- اضغط علي أيقونة تكوين القوائم أو اختار **menu editor** من قائمة **Tools**.



٢- ستظهر لك شاشة لتكوين القوائم. وتحوى علي العناصر التالية:

Caption: يستخدم هذا العنصر لكتابة بيانات القائمة، مثلاً بجميع القوائم الخاصة ببرنامج ويندوز نجد قائمة ملف **File** وبها نجد أمر حفظ **Save**، هنا **Caption** يمثل عنوان للقائمة تلك الاختيارات.

Name: اسم القائمة والذي يستخدم في برمجة كل القائمة أو عناصرها من خلال هذا الاسم، حيث يصبح الاسم هو المستخدم في الأمر المطلوب عند اختيار القائمة، أما العنصر **Caption** فهو خاص بالمستخدم وهو الذي يعرض علي الشاشة.

ويفضل أن يكون الاسم المطلوب كتابته داخل صندوق **Name** مطابقاً للعنوان الذي يكتب بالصندوق **Caption** مما يساعد علي تذكر اسم القائمة أثناء البرمجة بطريقة أسرع، ويجب أن يكون الاسم بدون فراغات داخله.

Index: تحتاج في بعض الأحيان إلى استخدام عدة اختيارات بالقائمة في نفس الوقت، يمكن هذا العنصر من إعطاء أرقام مسلسلة للاختيارات المتاحة بالقوائم وبالتالي تستطيع

الإشارة إلى هذه الاختيارات باستخدام أرقامها، مما يساعد علي الإشارة إلى الاختيارات من خلال مدى من الأرقام، لنفترض انك استخدمت هذا العنصر بالشكل التالي:
اختيار ١، اختيار ٢، اختيار ٣، اختيار ٤، فإذا أردت استخدام هذه الاختيارات حدد نطاقها من ١ إلي ٤ فيتم اختيارها في نفس الوقت، ولكن لو أن الاختيارات أخذت أسماء مختلفة سيكون عليك الإشارة إلى كل اختيار بالاسم المعطى له سابقاً.

Short Cut: يمكن اختيار حروف اختزال مباشرة من لوحة المفاتيح لاختيار أمر بالقائمة أو اختيار القائمة نفسها، في برنامج وورد مثلاً يمكن استخدام أمر الطباعة **Print** من قائمة الملف **File** أو بالضغط علي مفتاحي **Ctrl + p** وكذلك يمكنك تحديد حروف اختزال للقوائم التي تقوم بتكوينها لتسريع الاختيار.

Window List: يمكن هذا العنصر من تحديد أمر عرض النوافذ المستخدمة حالياً، مثلاً عند استخدام برنامج وورد، إذا كنت تستخدم أكثر من مستند فيمكنك استخدام قائمة إطارات **windows** للتنقل بين المستندات المستخدمة، ولا يعمل هذا الأمر إلا إذا كنت تستخدم برنامج يتألف من عدة إطارات، وبما أن البرنامج الذي سنستخدمه لا يستعمل أكثر من شاشة، فإن هذا الأمر **Window List** لن يعمل به.

Helpcontextld: يمكنك باستخدام لغة فيجوال بيسك تطوير برامج مساعدة بالبرامج التي تصممها، ويعتمد استدعاء ملف المساعدة علي المكان الذي تختاره، فلو اخترت قائمة **File** فإن ملف المساعدة يفترض أن يفتح صفحة المساعدة الخاصة بملف **File**، وهكذا بالنسبة للقوائم الاخري حيث يفترض أن يكون لها صفحات خاصة بها، ويرمز لكل صفحة برقم، ثم يتم ربط هذا الرقم مع القائمة أو الاختيار بالقائمة.

Checked: لوضع إشارة إلى جانب بعض الاختيارات بالقوائم لكي نعرف إذا كنا قد استخدمنا هذا الاختيار أم لا.

Visible: لاخفاء بعض الاختيارات أو القوائم مؤقتاً من علي الشاشة أو بالقائمة.

Ok: استخدم هذا الزرار بعد الانتهاء من تكوين أو تعديل القائمة، حيث يتم بعد ذلك إظهار القائمة على شاشة الواجهة مباشرة.

Cancel: لإلغاء التعديل أو التكوين.

الأسهم: استخدم الأسهم لتغير مستوى القوائم أو الاختيارات.

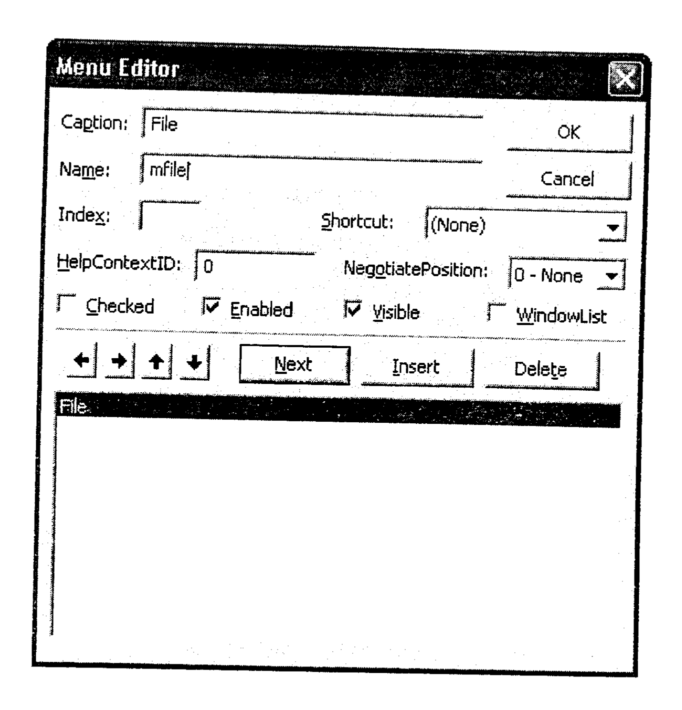
Next: لإضافة قائمة أو اختيار بالقوائم أو الاختيارات التي كونت مسبقاً.

Delete: لإلغاء قائمة أو اختيار.

بعد أن ظهرت لك شاشة تكوين القائمة تابع الخطوات التالية لتكوين عناصر القائمة:

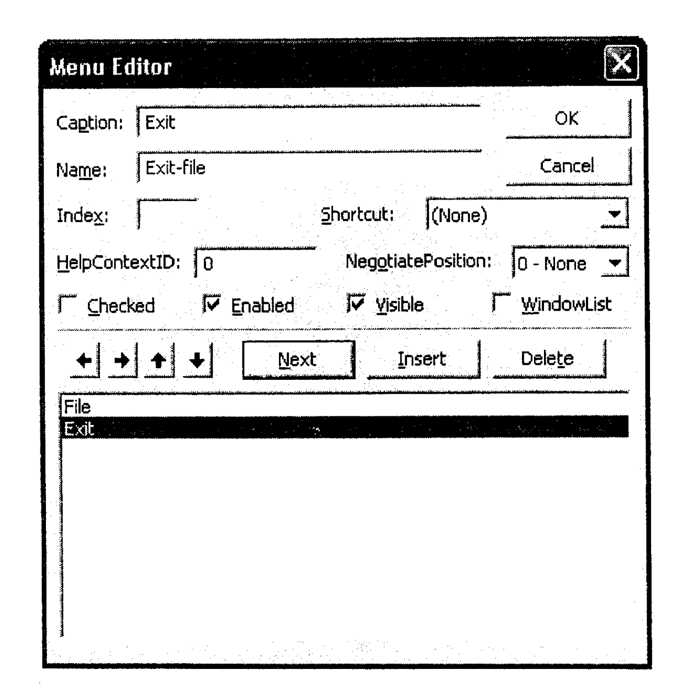
٣- بالصندوق **Caption** اكتب **File**.

٤- بالصندوق المعنون **Name** اكتب **mfile** ثم اضغط مفتاح الإدخال.



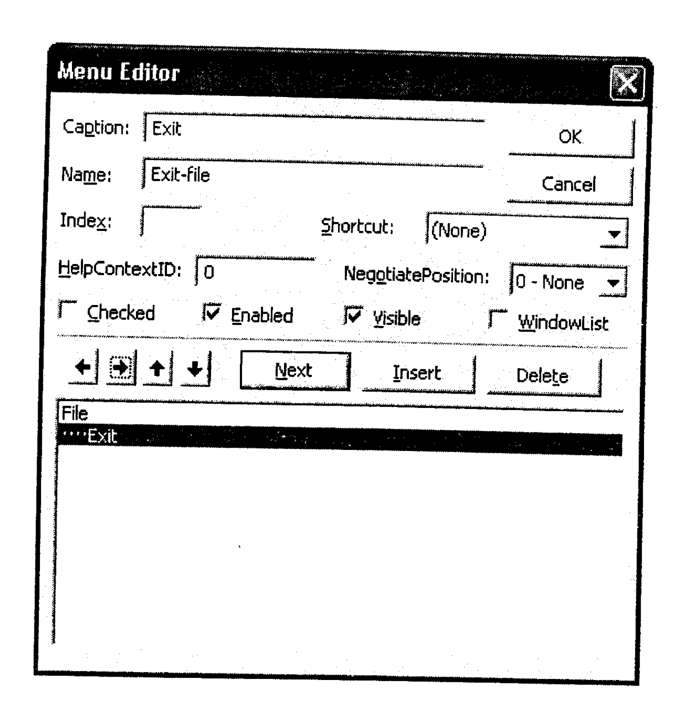
٥- اضغط زرار **Next** لمتابعة وكتابة الاختيارات الخاصة بالقائمة **File**.

٦- بالصندوق **Caption** اكتب **Exit**.



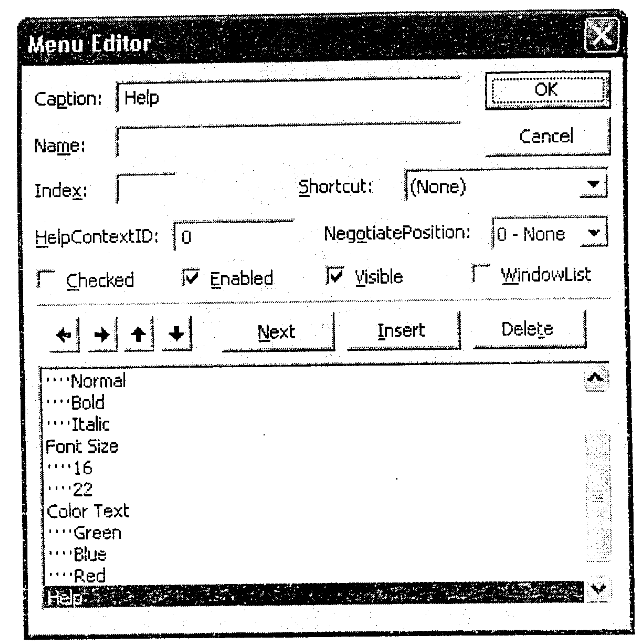
٧- بالصندوق **Name** اكتب **file-exit.**

٨- اضغط السهم الأيمن، هذا يعنى أن **Exit** عبارة عن اختيار من القائمة **File** وستلاحظ ظهور مسافة في بداية السطر، وهذا يعنى انك قد كونت اختيار تابع إلى القائمة، فالقوائم تبدأ من أول السطر، وإذا كتبت بيانات فـإن تحديـدها كقوائم أو اختيارات يعتمد علي مكان كتابتها بالنسبة لبداية السطر.



٩- تابع تكوين القوائم والاختيارات كما يلي:

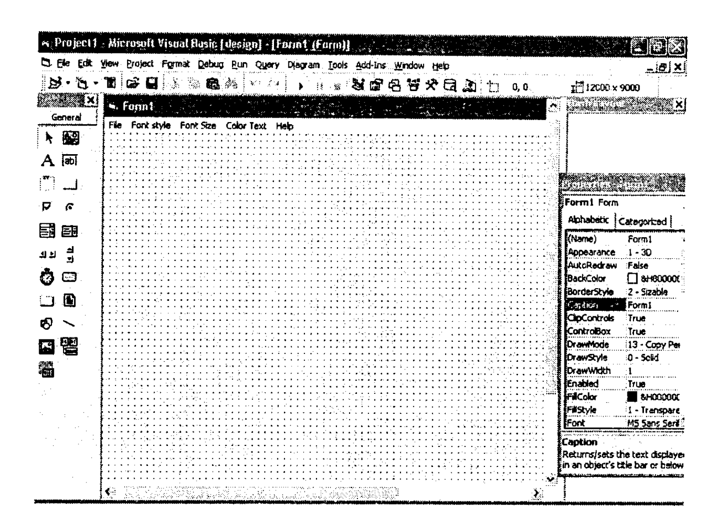
File
Exit
Font Style
…..Normal
…..Bold
…..Italic
…..Font Size
…….. 16
…….. 22
Color Text
……. Red
Help
….. A



الشكل التالي خاص بأسلوب ترتيب القوائم والاختيارات:

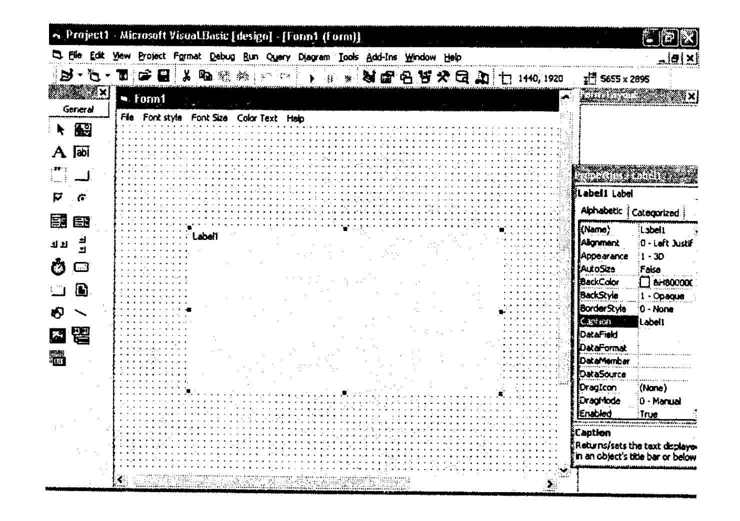
Caption	Name
File	Mfile
Exit	Exit
Font style	Font_style
Normal	Normal
Bold	Bold
Italic	Italic
Font Size	Font_size
16	Size 16
22	Size 22
Color Text	Color_text
Green	Green
Blue	Blue
Red	Red
Help	Help

الشكل التالي يظهر القوائم بأعلى النموذج:

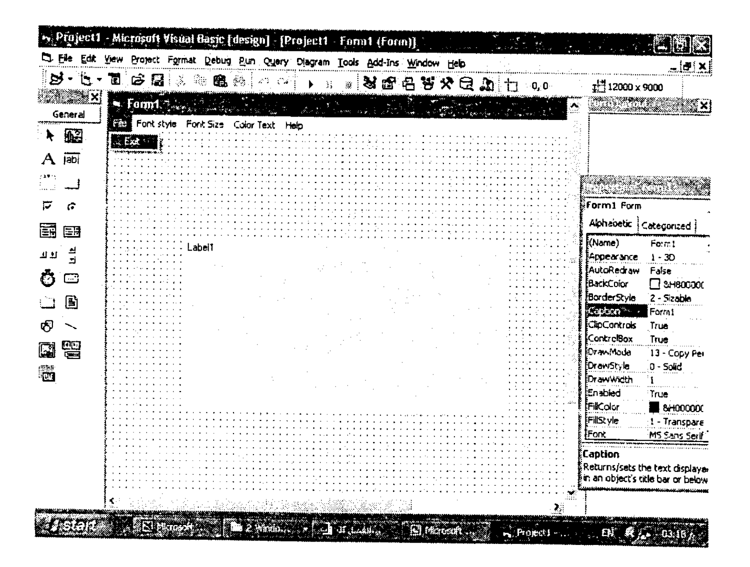


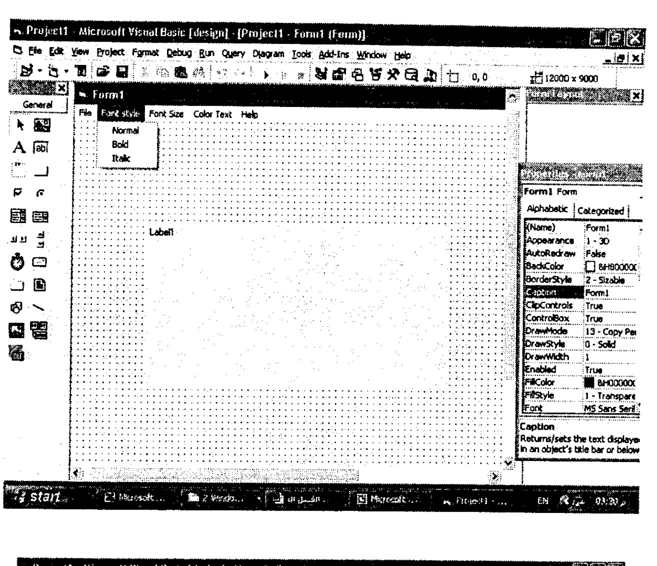
كتابة البرنامج:

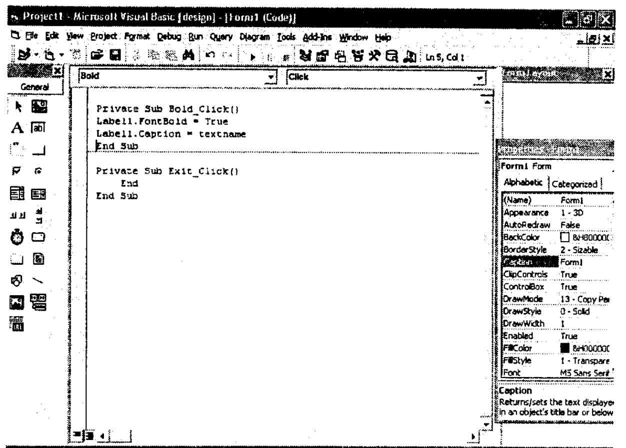
قبل كتابة البرنامج عليك إضافة كائن **Label** بوسط شاشة البرنامج وذو مساحة كافية لعرض البيانات.

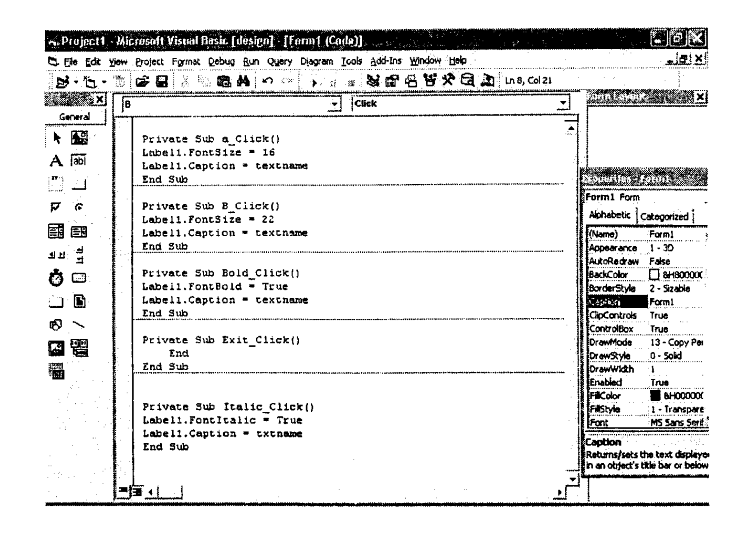


ثم اكتب الأوامر التالية:

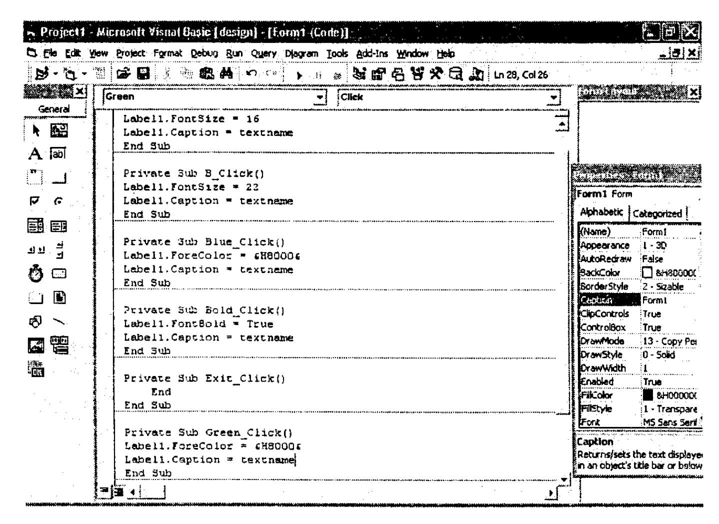



Project1 - Microsoft Visual Basic [design] - [Project1 - Form1 (Form)]
File Edit View Project Format Debug Run Query Diagram Tools Add-Ins Window Help
General
Form1
File Font style Font Size Color Text Help
Normal
Bold
Italic
Label1
Form1 Form
Alphabetic Categorized
(Name) Form1
Appearance 1 - 3D
AutoRedraw False
BorderStyle 2 - Sizable
Caption Form1
ClipControls True
ControlBox True
DrawStyle 0 - Solid
DrawWidth 1
Enabled True
FillStyle 1 - Transparent
Font MS Sans Serif
Caption
Returns/sets the text displayed in an object's title bar or below
start

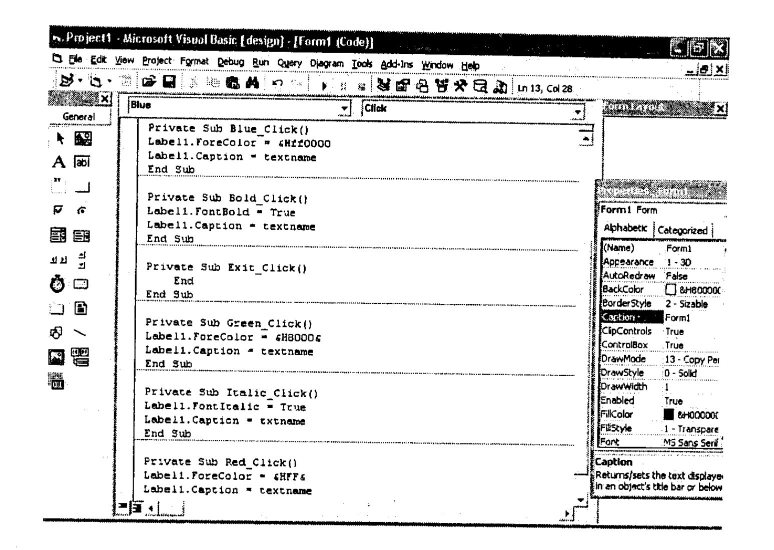

Project1 - Microsoft Visual Basic [design] - [Form1 (Code)]
File Edit View Project Format Debug Run Query Diagram Tools Add-Ins Window Help
Ln 5, Col 1
General
Bold
Click
Private Sub Bold_Click()
Label1.FontBold = True
Label1.Caption = textname
End Sub
Private Sub Exit_Click()
End
End Sub
Form1 Form
Alphabetic Categorized
(Name) Form1
Appearance 1 - 3D
AutoRedraw False
BorderStyle 2 - Sizable
Caption Form1
ClipControls True
ControlBox True
DrawStyle 0 - Solid
DrawWidth 1
Enabled True
FillStyle 1 - Transparent
Font MS Sans Serif
Caption
Returns/sets the text displayed in an object's title bar or below

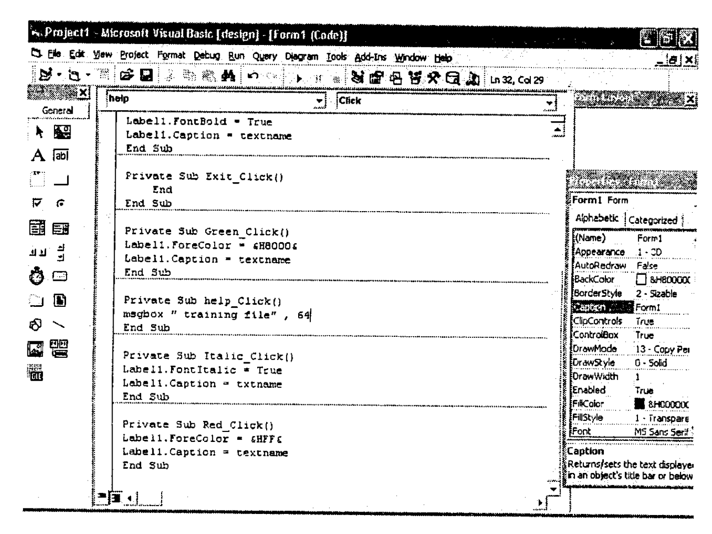


Project1 - Microsoft Visual Basic [design] - [Form1 (Code)]
Private Sub a_Click()
Label1.FontSize = 16
Label1.Caption = textname
End Sub
Private Sub B_Click()
Label1.FontSize = 22
Label1.Caption = textname
End Sub
Private Sub Bold_Click()
Label1.FontBold = True
Label1.Caption = textname
End Sub
Private Sub Exit_Click()
End
End Sub
Private Sub Italic_Click()
Label1.FontItalic = True
Label1.Caption = txtname
End Sub



Project1 - Microsoft Visual Basic [design] - [Form1 (Code)]
Label1.FontSize = 16
Label1.Caption = textname
End Sub
Private Sub B_Click()
Label1.FontSize = 22
Label1.Caption = textname
End Sub
Private Sub Blue_Click()
Label1.ForeColor = &H8000&
Label1.Caption = textname
End Sub
Private Sub Bold_Click()
Label1.FontBold = True
Label1.Caption = textname
End Sub
Private Sub Exit_Click()
End
End Sub
Private Sub Green_Click()
Label1.ForeColor = &H8000&
Label1.Caption = textname
End Sub






```
Dim textname As string
Sub Blue_click ( )
Labell.Forecolor = &HFF0000
Labell.caption = txtname
End Sub
```

```
Sub Bold_Click ( )
Labell.FontBold = True
Label1.caption = Txtname
End Sub

Sub File_exit_click ( )
End
End Sub

Sub Form_load ( )
Txtname = "الفيومى "
Label1.Alignment = 2
Label1.Caption = txtname
End Sub

Sub green_Click ( )
Label1.Forecolor = &H8000&
Labell.Caption = txtname
End Sub

Sub help_Click ( )
Msgbox "ملف تدريبي ",64
End Sub

Sub Italic_Click ( )
Label1.Fontitalic =True
Label1.Caption = txtname
End sub

Sub Normal_Click ( )
Label1.Fontitalic = False
Label1.FontBold = False
Label1.Caption = txtname
End Sub
```

```
Sub red_Click ( )
Label1.Forecolor = &Hff&
Label1.Caption = txtname
End Sub

Sub size_10_Click ( )
Label1.Fontsize = 10
Label1.Caption = txtname
End Sub

Sub size_14_Click ( )
Label1.Fontsize = 14
Label1.Caption = txtname
End Sub

Sub Size-16_Click ( )
Label1.Fontsize = 16
Label1.Caption = txtname
End Sub

Sub size_18_Click ( )
Label1.FontSize = 18
Label1.Caption = txtname
End Sub

Sub size_22_click ( )
Label1.Fontsize = 22
Label1.Caption = txtname
End Sub

Sub Text1_Click ( )
Label1.Caption = txtname
End Sub
```

وما يلي شرح للأوامر:

Labell.Forecolor: لتغيير لون الكتابة.

Label1.bold: لتغيير نوع الكتابة بحروف ثقيلة.

Label1.Fontsize: لتغيير حجم الكتابة.

MsgBox: لإظهار رسائل تظهر علي الشاشة بصناديق. رقم ٦٤ يمثل الإشارة الـتي ستظهر بالصندوق. وهذه الإشارات تجدها بملف المساعدة، حيث يمكنك طلب البحث ثم اختار البحث عن **Msgbox**، سيوضح لك هذا الملف الأرقام المقابلة التي يمكن إظهارها بهذا الصندوق.

ملاحظة: تأكد من كتابة **Dim Txtname** في الجزء المعنون **General**.

كتابة رموز الألوان:

تستخدم لغة فيجوال بيسك رموز غير رقمية يصعب تذكرها، وإذا رغبت في اســتخدام الألوان بالبرنامج، ولمعرفة اللون المطلوب، اختار شكل بالشاشة ثم اضغط مفتــاح **F4** واعرض اللون المطلوب، سيظهر لك رقم اللون في اعلي الشاشة. مــن شاشــة تحديــد المواصفات الخاصة بالشكل الذي اخترته، اكتب هذا الرقم علي ورقة ثم اكتبــه داخــل البرنامج.

ويمكن استخدام إحدى الطرق التالية للتحكم في الألوان.

١- اســـتخدام ألـــوان **VB** مثـــل **VBRED**، **VBBLUE**، **VBBLACK**، **VBWHITE**.

٢- استخدام أرقام الألوان الرئيسية بأمر:

RGB (أزرق ,أخضر ,أحمر)

ونحدد درجة كل لون من ٠ حتى ٢٥٥.

٣- استخدام ألوان **Quick Basic** بالأمر

QBCOLOR (رقم)

ونكتب رقم اللون من ٠ إلى ١٥.

الرسم في لغة الفيجوال بيسك

تستخدم أوامر الرسم في **vb** لرسم نقط **Pset** أو خطوط **Line** أو دوائر **Circle**

وتوجد عدة طرق لتحديد الألوان منها :

أولا: ألوان الفيجوال بيسك

vbcolor

مثل

vbwhite, vbred, vbblack, vbblue

وغيرها من الألوان

ثانيا: في بعض الأحيان تحتاج لدرجة لون محددة وهنا نستخدم وظيفة **RGB Function** لتحديد قيمة الألوان الأساسية (احمر ، اخضر ، ازرق) بأستخدام الامر التالى:

RGB (0, 0, 0)

يمكن إعطاء قيمة الألوان من ٠ إلى ٢٥٥

أمثلة :

```
FORM1.BACKCOLOR = RBG (0,128, 0)     '(Green)
FORM2.BACKCOLOR = RGB (255 , 255, 0 )   '
(Yellow)
Pset ( 100,100), RGB ( 0,0,64)    ' Set Point to blue
dark
```

ثالثا: يمكن استخدام وظيفة ألوان كويك بيسك **QBCOLOR**

وهى تحدد قيمة الألوان طبقا لجداول الألوان الموجودة في ميكروسوفت كويك بيسك وهى الألوان من ٠ إلى ١٥.

مثال :

```
FORM1.BACKCOLOR = QBCOLOR (5)
```

من الصيغ الأساسية للتعامل مع الألوان الدالتين التاليتين وهما عكس بعضهما:

```
Pset( , )
point( , )
```

الدالة **Pset** :

تقوم بإرسال لون لنقطة محددة مثلا

```
pset(100,100)
```

ترسل إلي النقطة ذات الإحداثيات (١٠٠,١٠٠) لونا هو لون الخط في النموذج:

```
form1.forecolor
```

والوضع الطبيعي هو اللون الأسود ما لم يغيره المستخدم بالنسبة للإحداثيات في الشاشة:
تعتبر النقطة أعلى يسار الشاشة هي النقطة (٠,٠) والتدرج يزداد كلما اتجهنا يمينا أو إلى أسفل
أي أن المحور الرأسي معكوس من ما هو معتاد لنا في الرسم البياني.
نفرض أننا نريد وضع نقطة بيضاء في النقطة (٥٠,١٥٠) فيصبح الكود

```
pset(50,150),vbwhite
```

ستلاحظ أن النقطة صغيرة جدا ويمكن التحكم في حجم النقطة عن طريق الأمر التالي

```
form1.drawwidth=7
```

ووضع أي رقم أكبر من الرقم ٧ الذي في المثال السابق.

الأمر **point**

يقوم بالعملية العكسية فباختيار أي نقطة تقوم هذه الدالة بإرجاع درجة لونها

مثال:

```
Private Sub Form_MouseDown(Button As
Integer, Shift As Integer, X As Single,
Y As Single)
form1.drawwidth=7
pset(x,y),rgb(100,100,100)
end sub
```

سيقوم البرنامج برسم خطوط متقطعة اقرب إلى نقاط متقطعة مع حركة الفارة

لاستخدام هذه الأساليب لابد من تحديد الكائن **object** ، وألا سيعتبر **VB** أن الكائن المراد الرسم عليه هو النموذج الحالي.

المثال التالي يرسم على النموذج **MyForm**

MyForm.pset(500, 500)

المثال التالي يرسم في صندوق صورة يسمى **PicShow**

PicShow.Pest (500, 500)

لمثال التالي يرسم في النموذج الحالي:

Pest (500 , 500)

أمر **CLS**:

يستخدم لمسح الشاشة:

PicShow.cls

أمر رسم الخط والشكل الرباعى **Line** :

Line(,) – (,)

وبإعطاء إحداثيات نقطتين يتم رسم خط بينهما مثلا مع إضافة

Text1 and text2 to the form

```
Private Sub Form_MouseMove(Button As
Integer, Shift As Integer, X As Single,
Y As Single)
Form1.ForeColor = vbRed
Line (X, Y)-(Val(Text1.Text),
Val(Text2.Text))
Text1.Text = X
Text2.Text = Y
End Sub
```

سأعدل في آخر مثال وقد كان يرسم خطوطا كلما تحرك الفارة ونود أن نتحكم في ذلك بألا يرسم ما لم يضغط المستخدم على الفارة

أولا نحتاج إلى متغير **global** و لنسمه **x1** (أي أنه متاح لكل جزئيات النموذج) نصرح عن المتغير في أعلى الكود في جزئية **General**

سنعطى هذا المتغير القيمة ١ إذا كان الفارة مضغوطا والقيمة ٠ في غير ذلك

```
Private Sub Form_MouseDown(Button As
Integer, Shift As Integer, X As Single, Y As
Single)
x1=1
end sub
Private Sub Form_Mouseup(Button As Integer,
Shift As Integer, X As Single, Y As Single)
x1=0
end sub
```

سنعيد نفس الكود السابق مع مراعاة التنفيذ إذا كان المتغير =١

ليصبح كل الكود كما يلي

```
dim x1 as integer
```

```
Private Sub Form_MouseDown(Button As
Integer, Shift As Integer, X As Single,
Y As Single)
x1=1
end sub

Private Sub Form_Mouseup(Button As
Integer, Shift As Integer, X As Single,
Y As Single)
x1=0
end sub

Private Sub Form_MouseMove(Button As
Integer, Shift As Integer, X As Single,
Y As Single)
Form1.ForeColor = vbRed
if x1=1 then
Line (X, Y)-(Val(Text1.Text),
Val(Text2.Text))
Text1.Text = X
Text2.Text = Y
End if
End Sub
```

ويستخدم أمر **line** لتوصيل خط من آخر نقطة تم رسمها لنقطة أخرى كما يلي

Line(1000,1000)-

تم رسم الخط من آخر نقطة تم رسمها (٥٠٠،٧٠٠) إلى النقطة (١٠٠٠،١٠٠٠) – لاحظ استخدام "–" كما يمكن رسم الأشكال الرباعية – المستطيل أو المربع – بالأمر **line** كما يلي

Line(500,500)-(1000,1000),,B

تم رسم شكل رباعي طول ضلعه ٥٠٠ (تويب) من النقطة (٥٠٠،٥٠٠) إلى النقط (١٠٠٠،١٠٠٠) ويمكن رسم نفس هذا الرسم باستخدام **step** والتي نستخدمها

لرسم النقط واحدة بالأخرى انظر المثال

```
Line (500,500) - step (1000, 0)
Line- step(0,1000)
Line - step( - 1000 , 0)
Line- step ( 0, -1000)
```

الخطوة الأولى باستخدام **step** رسمنا خط طوله ١٠٠٠ تويب من عند النقطة ٥٠٠ موازيا للمحور **x**

الخطوة الثانية باستخدام **Step** رسمنا خط طوله ١٠٠٠ تويب من عند النقطة ٥٠٠ موازيا للمحور **y** عموديا على الخط الأول. وهكذا .. رسمنا الشكل الرباعى باستخدام **step**

لاحظ علامتي ",," فهما يستخدما لإدراج اللون ، كما أن استخدام حرف **b** يستخدم لرسم شكل رباعي مفرغ ، ولرسم شكل راعى مصمت نستخدم **bf** كما يلي

```
line(500,500)-(1000,1000),qbcolor(4),bf
```

رسمنا شكلا رباعيا مصمتا لونه احمر .

أمر رسم الدوائر **Circle** :

```
[Object.] Circle[Step] (x,y), Radius [ , Color]
```

المثال التالي يرسم دائرة نصف قطرها **radius** عشوائي وبألوان عشوائية:

```
Sub CircleDemo( )
Dim Radius
R = 255 * Rnd
G = 255 * Rnd
B = 255 * Rnd
Xpos = ScaleWidth/12
Ypos = ScaleHeight/12
Radius = (( YPOS * 0.9) +1) * Rnd
```

```
Circle ( Xpos,Ypos ،Radius) , RGB(R,G,B )
END SUB
```

لرسم القوس:

لابد من تحديد الزاوية أو نقطة البداية والنهاية

```
[ Object.] Circle[Step] (x,y), Radius, [color],
Start,End, [aspect]
```

- قيمة **aspect** هي النسبة بين البعد العمودي والبعد الافقي ويمكن رسم الشكل البيضاوي كما في المثال التالي:

```
Private sub form-click( )
FillStyle = 0
Circle ( 600, 1000) , 800 ,,,, 3
FillStyle = 1
Circle ( 1800 , 1000) ,1/3,,,,800
END SUB
```

لاحظ استخدام **fillstyle** ، فإذا كانت قيمتها = صفر فأن الرسم يكون مفرغا ، وإذا كان يساوى واحد فأن الرسم يكون مصمتا

- لرسم الدائرة

```
circle(x,y),(100)
```

حيث البارمترين الاولان للمركز والعناصر الثالث لنصف القطر
يمكنك أن تحدد لون المحيط بالصيغة

```
form1.forecolor=(color)
```

أو

```
circle(x,y),(200),vbwhite
```

أما لون التعبئة فبالصيغة

```
form1.fillcolor=(color)
```

ولإضافة تعبئة فعليك أن تختار واحدة من ثمانية أنواع للتعبئة

٠ وهي تعبئة كاملة

١ بدون تعبئة أو شفاف

٢ خطوط أفقية

٣ خطوط راسية

٤–٥ خطوط مائلة

٦ خطوط متقاطعة

٧ خطوط متقاطعة مائلة

لتعبئة خطوط متقاطعة مثلا نستعمل الصيغة

```
form1.fillstyle=6
```

يمكن إدراج جزء من الدائرة وذلك بإضافة عدد بين ٠ إلى ٦ بعد صيغة الرسم للدائرة وذلك كالآتي

```
circle(x,y),(100),,3
```

لحفظ الرسم نستخدم الصيغته التالية:

```
SavePicture   Form1.Image,"C:\flower.bmp
"
```

نلاحظ انه يقبل بارمترين..الثاني نكتب فيه المسار واسم الرسم مع الامتداد

إذا أردت إعطاء المستخدم حرية مسار الحفظ نستعمل من **components** الأداة **ms common dialog** مع استعمال الصيغة.**Save** في كود الحفظ

أضف جملتين لأحد الاكواد المكتوبة لتزيد من إمكانية البرنامج

```
dim x1 as integer
Private Sub Form_MouseDown(Button As
Integer, Shift As Integer, X As Single,
Y As Single)
Text1.Text = X
Text2.Text = Y
x1=1
end sub

Private Sub Form_Mouseup(Button As
Integer, Shift As Integer, X As Single,
Y As Single)
x1=0
end sub

Private Sub Form_MouseMove(Button As
Integer, Shift As Integer, X As Single,
Y As Single)
Form1.ForeColor = vbRed
if x1=1 then
Line (X, Y)-(Val(Text1.Text),
Val(Text2.Text))
Text1.Text = X
Text2.Text = Y
End if
End Sub
```

لاحظ أننا أضفنا قيم ل **text1 , text2** داخل كود أل **MouseUp** بحيث يمكن رسم خطوط منفصلة كلما ضغطنا الفارة ضغطة جديدة

حاول أن ترسم أي شيء ثم قم بتصغير النموذج **minimize** ثم أعده ستلاحظ أن الرسم قد اختفى ...لتحاشي اختفاء الرسم أضف السطر التالي في أل**form_load**

```
Form1.AuteRedraw=True
```

لننتقل إلى مرحلة أخرى وهي التحكم بصور موجودة أصلا في الجهاز

المثال الأول سندرج صورة إلى النموذج ثم نقوم بنسخ صورة منها على نفس النموذج لنسمي الصورة **image1** ثم نستخدم الكود التالي

```
Private Sub Command1_Click()
Dim x
Image1.Stretch = True
Image1.Move 0, 0, 2000, 1000
For i = 0 To 2000 Step 10
For j = 0 To 1000 Step 10
x = Point(i, j)
PSet (i, j + 2000), x
Next
Next
End Sub
```

كل ما تحتاجه هنا **command** و**image**

وإذا أردنا نسخ الصورة بشرط أن تكون النسخة معكوسة نستعمل الكود السابق مع التعديل التالى:

```
Private Sub Command1_Click()
Dim x
Image1.Stretch = True
Image1.Move 0, 0, 2000, 1000
For i = 2000 To 0 Step -10
For j = 0 To 1000 Step 10
x = Point(i, j)
PSet (2000 - i, j + 2000), x
Next
Next
End Sub
```

إنشاء رسم متحرك Animation

يمكن إنشاء رسم متحرك وذلك بالتبادل في إظهار رسمين أو أكثر كما يمكن ذلك بتحريك

الصورة :

- ضع أداة الصورة PICTURE على سطح النموذج ، غير الاسم إلى picIcon
- انسخ هذه الأداة والصقها بالنموذج ، ستظهر رسالة لإنشاء مصفوفة من أداة الصورة picIcon
- كرر ذلك ثانية
- أصبح لديك الآن ثلاث أدوات لصندوق الصور هي

picIcon(0), picIcon(1), picIcon(2)

- ضع صندوق صورة على النموذج وغير اسمها إلى picDisp
- من قائمة الخواص حدد الأيقونة التي تظهر في كل من صناديق الصور الثلاث الأولى وهي صور إشارة المرور الخضراء والانتظار الصفراء والحمراء بالتوالي وستجد هذه الصور في مكتبة الصور الواردة مع vb6 على مجلد

common/graphics/icons/traffic

- ضع أداة المؤقت timer
- في حدث timer1_timer اكتب الأوامر

```
Private Sub Timer1_Timer( )
If Form1.picDisp = PicIcon1(0).Picture Then
Form1.picDisp = PicIcon1(1).Picture
ElseIf Form1.picDisp = PicIcon1(1).Picture Then
Form1.picDisp = PicIcon1(2).Picture
ElseIf Form1.picDisp = PicIcon1(2).Picture Then
Form1.picDisp = PicIcon1(0).Picture
End If
End Sub
```

جرب تشغيل هذا البرنامج، هل رأيت الإشارات الثلاث يتبادلن الظهور في صندوق الصور picDisp ، إذا حدث ذلك فقد نجحت وألا اعد كتابة البرنامج لا تنسى أن

تكتب الأمر التالي في form1_load

```
Form1.picDisp = PicIcon1(0).Picture
```

جرب هذا البرنامج:

ضع أداة المؤقت timer، اضبط مواصفة interval =500 أو أكثر إذا أردت أن تجعل الاستجابة بطيئة واكتب الأوامر التالية

```
Private Sub Timer1_Timer( )
Dim OldDrawStyle %
OldDrawStyle% = Form1.DrawStyle
Form1.DrawStyle = vbSolid
DrawRubberBand picDisp.Top, picDisp.Left ،
picDisp.Height, picDisp.Width
Form1.DrawStyle = OldDrawStyle %
End Sub
```

- اكتب الإجراء التالي

```
Sub DrawRubberBand(ByVal x1, ByVal y1,
ByVal x2, ByVal y2 As Single)
Dim OldDrawMode %
OldDrawMode% = Form1.DrawMode
Form1.DrawMode = vbInvert

Line (y1 - 10, x1 - 10)-(y2 + y1 + 10, x2 + x1 +
10), , B
Form1.DrawMode = OldDrawMode %
End Sub
```

لاحظ استخدام أوامر drawmod ,drawstyle

استخدام **AutoRedraw** :

خاصية **AutoRedraw** هي خاصية **boolean** والتي بضبطها إلى **TRUE** يجعل مخرجات الرسم تخزن في ذاكرة الحاسب .

فعند عرض نافذة فوق نافذة أخرى، يجب حفظ النافذة الأولى بالذاكرة حتى يمكن إعادتها ثانية بعد انتهاء استخدام النافذة الثانية، لذا يجب أن تضبط خاصية **AutoRedraw** للأولى إلى **TRUE** والثانية إلى **FALSE** حتى يمكن إلغاؤها.

لنفرض أننا نريد أن نرسم دائرة سوداء تتحرك نستعمل التايمر ونجعل خاصية ال**enabled = false** وخاصية **interval=50** ثم نضيف زر للنموذج أضف الكود التالي

```
Dim x As Integer
Private Sub Command1_Click()
Timer1.Enabled = True
End Sub

Private Sub Timer1_Timer()
Cls
x = x + 100
Circle (x, 5000), (500)
End Sub
```

الامر **CLS** يقوم بمسح محتويات الشاشة من رسومات وكتابة داخل كود التايمر لاحظ الآتي أولا نمسح محتويات الشاشة ثانيا المتغير **X** وهو الاحداثي السيني لمركز الدائرة نقوم بتغييره بمقدار معين ونرسم الدائرة الجديدة بعد تغيير الاحداثي السيني لاحظ انه يمكن تغيير كل من الأرقام والاحداثي الصادي لنجعل التحرك في اتجاهات مختلفة.

كيفية تحريك الاشياء فى فيجوال بيسك

أولا يجب أن نعرف بعض من الاشياء التى يمكن تحريكها وهى :

١. **label** وهو عبارة عن ورقة تخيلية مرئية بواسطة **visual basic** نستطيع الكتابة عليها لتوضيح وظيفة معينة .
٢. **image** وهى صورة يمكن فى الفيجوال بيسك من اضافتها للبرامج المصممة عن طريقه وتظهر بشكل مستوى على الفورم اى كأنها مرسومة على الفورم.
٣. **picture** وهى ايضا صورة وهى تماما مثل السابقة الا انها تختلف عنها فى طريقة او شكل اظهار الفيجوال بيسك لها حيث تكون مغمورة داخل الفورم
٤. **button** وهذه الاداة نراها كثيرا حيث لا غنى عنها فى اى برنامج حتى ان نظام الويندوز قائم عليها وهى الزرار.

طريقة التحريك:

نفتح البرنامج ونختار **form.exe** فتظهر لنا واجهة المشروع او الويندو او الفورم نختار الاداة التى نريد تحريكها من شريط الادوات او الـ **toolbox** ولتكن **label**

بعد اضافة بواسطة النقر عليه بالزر الايسر بالفارة مرتين على الاداة وضبط مساحتها نلاحظ ان تلك الاداة مكتوب عليها **label1** وهذا اذا كانت اول مرة تضاف الى الفورم ومعنى هذا انه اذا تم اضافتها مرة اخرى الى الفورم فسنجدها مكتوب عليها **label2**وهكذا ولتغيير او لكتابة اى شئ على هذة الاداة نقوم بالذهاب الى قائمة الخصائص او الـ **properties** وهى موجودة على يمين الصفحة ونختار منها الخاصية **caption** ونكتب الاسم او الجمل التى نريد كتابتها بداخل الخانة الموجودة امام خانة الـ **caption**

ولتحريك اى شئ فى بيئة الفيجوال بيسك لا بد من اختيار الاداة المسؤلة غن ذلك وهى الـ **timer**

ونقوم بإضافتها من الـ **toolbox** وعند أضافتها فستظهر قائمة خصائص التوقيت فى الجهة اليمنى مكان خصائص الفورم واذا قمنا بالضغط على اى مكان من سطح الفورم

فستظهر خصائص الفورم مرة اخرى ولكن لا تفعل ذلك قبل ان نكتب بعض الاشـــياء بداخل خصائص التوقيت الا وهى القيمة التى سنضيفها فى مربع الخاصية التى تســمى **interval**وتكتب هذه القيمة بالمربع الموجود امام هذة الخاصية ولتكن هذة القيمة **40**وتعبر هذة القيمة عن عدد حجم الخطوة التى يقطعها الــ **label**

وبعد كتابة هذة القيمة نذهب الى قائمة الاوامر لكتابة الكود الخاص بالحركة ويكــون بداخل الجزء الخاص بالــ **timer** كما يلى :

١. اذا اردنا تحريكة ناحية اليمين فيكون الكود على النحو التالى

```
Private Sub Timer1_Timer()
Label1.Left = Label1.Left + 40
End Sub
```

٢. اذا اردنا تحريكة ناحية اليسار فيكون الكود على النحو التالى

```
Private Sub Timer1_Timer()
Label1.Left = Label1.Left  40
End Sub
```

٣. اذا اردنا تحريكة الى اعلى فيكون الكود على النحو التالى

```
Private Sub Timer1_Timer()
Label1.Top = Label1.Top - 40
End Sub
```

٤. اذا اردنا تحريكة الى اسفل فيكون الكود على النحو التالى

```
Private Sub Timer1_Timer()
Label1.Top = Label1.Top + 40
End Sub
```

الرسم البياني تطبيق مهم جدا حيث يعطي تصورا وملخصا عن الحقول في السـجلات وبالنسبة لفيجوال بيسك فهو يتيح لك عمل الرسوم البيانية التي تستمد مصدرها مـن قاعدة بيانات.

هناك عدة طرق لإظهار الرسم البياني ، فيمكنك عرض رسم يبين القيم في سجل واحد مثلا درجات طالب في مادتين ولكي ترى الرسم البياني الذي يمثل الطلاب الاخرين تنتقل بزرار ينقلك للسجل التالي.

أو تقوم بعرض جميع سجلات الطلاب في رسم واحد مقسم على اساس كل سجل.

قبل ان تبدأ يجب ان تقوم بعمل قاعدة بيانات في اكسس اسمها **db1** تحتوي على جدول اسمه **tb1** فيه الحقول التالية:

إسم الحقل	نوع البيانات
name	**String**
Mathmark	**Integer**
Sincemark	**Integer**

أضف سجلات لقاعدة البيانات بالقيم التي تريدها

افتح مشروع فيجوال بيسك جديد ، ثم عرف او أربط قاعدة البيانات في البرنامج بالكود

اذهب الى **Project >> references**
ثم حدد المكتبة **Microsoft DAO 3.51 Object library**

عرف المتغيرين في قسم الاجراءات العامة كما في الكود التالي:

```
Public d As Database
Public tb As Recordset
```

اكتب الكود التالي والذي يقوم بربط قاعدة البيانات بالبرنامج

```
Set db =
DBEngine.Workspaces(0).OpenDatabase(App
.Path & "\db1.mdb", True)
Set tb = db.OpenRecordset("tb1",
dbOpenTable)
```

الخطوة التالية هي اضافة أداة الرسم البياني ولذلك اذهب قائمة **Project** واختر **Components**

سيظر لك مربع حوار حدد منه الاختيـــار **Microsoft Chart Control 6.0 (OLEDB)**

ستظهر لك الايقونه الخاصة به في صندوق الادوات بعد ذلك انقر على هذه الايقونة وارسم شكل مربع على اغلب الفورم وسينتج لك صورة رسم بياني.

حدد الرسم البياني واذهب الى نافذة الخائص وغير الخاصية **AutoIncrement** إلى **True** وهذا ضروري لكي يقوم الرسم البياني بتغيير وتحديث نفسه تلقائيا ولن يعمل بدون ان تجعل هذه الخاصية **True**

اضبط الخاصية **ColumnCount** على ٢ لتحديد عدد الاعمدة في الرسم البياني وبما اننا نحتاج الى عمودين فقط هما درجة الرياضيات والعلوم وضعنا عدد الاعمدة ٢ .

اضبط الخاصية **RowCount** على ١ وهذا الرقم يدل على عدد السجلات التي تريد ان تعرضها في المرة الواحدة.

ماهو الكود عرض البيانات على الرسم البياني ، اولا يجب ان تحدد متى يتم العرض هل عند تشغيل البرنامج ام عند النقر على زرار معين ؟ والطريقة الافضل ان تجعله يعرض وقت تنفيذ او تحميل الفورم وبعد ذلك يمكنه ان ينتقل الى السجل التالي بواسطة زرار تنقل ، لذا يجب وضع كود في حدث التحميل للفورم لكي يعرض مباشرة والكود سيكون كما يلى:

لاحظ ان اسم ادة الرسم البياني هى `MSChart1`

```
Private Sub Form_Load()
Set db =
DBEngine.Workspaces(0).OpenDatabase(App
.Path & "\db1.mdb", True)
Set tb = db.OpenRecordset("tbl",
dbOpenTable)

MSChart1.Column = 1
MSChart1.Data = tb!mathmark
MSChart1.Column = 2
MSChart1.Data = tb!sincemark
MSChart1.RowLabel = tb!Name

End Sub
```

هذا الكود يذكرنا بالأداة **FlexGrid** ، حيث حددنا اولا رقم العمود الذي ستنفذ عليه الاوامر القادمة وهو عمود رقم ١ ثم بعد ذلك وضحنا الامر الذي نريده ان ينفذ على هذا العمود وهو وضع قيمة الحقل **mathmark** في هذا العمود ليعبر عنها بيانيا ونستخدم لهذا الامر **Data** كما هو واضح ، ولوضع قيم الحقل الثاني وهو **sincemark**في العمود الثاني، نتبع نفس الخطوات السابقة حيث نحدد رقم العمود وهو ٢ ثم ما نريد منه وهو التعبير عن هذا الحقل في شكل بياني.

بعد ذلك وفي السطر الاخير نريد ان يضع البرنامج اسم الشخص اسفل الرسم البياني وهو ما يسمى بالصف **Row** ونستخدم الخاصية **RowLabel** ونخزن فيها قيمة حقل الاسم **Name**

المرحلة الثانية وهي ان يقوم المستخدم بالتنقل بين الحقول لإظهارها في شكل الرسم البياني وسنقوم بإنشاء زرارّي أمر، واحد للتقدم للأمام وواحد للرجوع للخلف ، في الزرار المخصص للتنقل للإمام نستخدم الكود التالي:

```
Private Sub Command1_Click()
tb.MoveNext   ' للإنتقال للسجل التالي
```

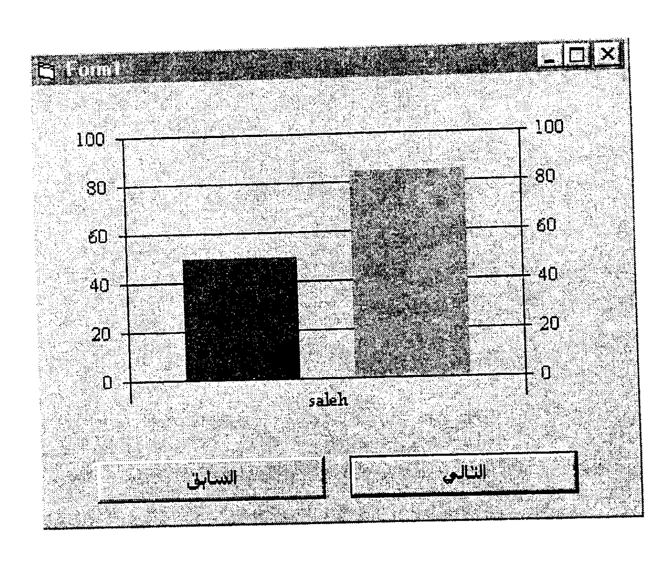
```
If tb.EOF Then tb.MoveLast'لتفادي المشاكل عند اخر سجل
MSChart1.Column = 1
MSChart1.Data = tb!mathmark
MSChart1.Column = 2
MSChart1.Data = tb!sincemark
MSChart1.RowLabel = tb!Name
End Sub
```

لاحظ أن الكود هو نفس الكود في حدث التحميل مع الاخذ في الاعتبار نقطتين أولا الانتقال للسجل التالي لكي يتم عرض البيانات في السجل التالي، ووضع شرط ليتحقق هل تم الوصول للسجل الاخير ليقوم بالخروج من الاجراء لكي نتفادى مشاكل توقف البرنامج.

أما الكود الذي يقوم بالرجوع الى الخلف فهو:

```
Private Sub Command2_Click()
tb.MovePrevious
If tb.BOF Then tb.MoveFirst
MSChart1.Column = 1
MSChart1.Data = tb!mathmark
MSChart1.Column = 2
MSChart1.Data = tb!sincemark
MSChart1.RowLabel = tb!Name
End Sub
```

الكود السابق هو نفس كود الانتقال للإمام مع تغيير امر التحرك للأمام بأمر التحرك للخلف MovePrevious وتم تغيير شرط التأكد من وصولنا للسجل الاول بحيث يتوافق مع الحاله



الشكل النهائي

لاحظ أن المحورين يتم تقسيمهما تلقائيا على حسب الارقام في السجلات

ويمكنك التحكم في بعض الخصائص مثل جعل الرسم البياني على شكل خطوط او مجسمات ثلاثية الابعاد وغير ذلك يمكنك اكتشاف ذلك بنفسك بالضغط على الزرار الايمن على الرسم البياني واختيار **Properties**

الفصل الثاني عشر

اكتشاف وتصحيح الأخطاء

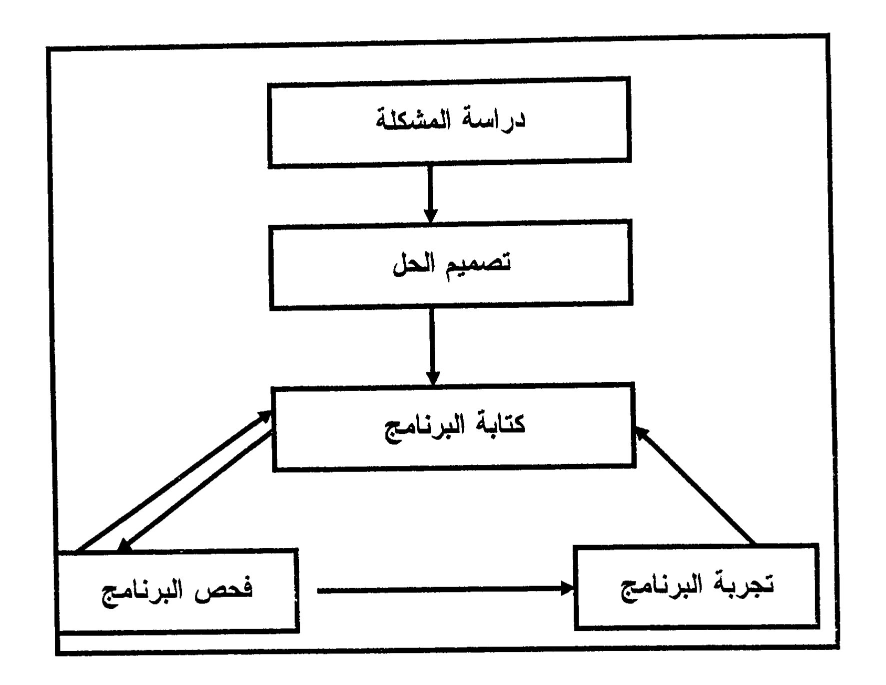
سندرس الأخطاء وتأثيرها علي عمل البرنامج، ثم كيفية استخدام لغة فيجوال بيسك في اكتشاف الأخطاء التي يمكن أن تحدث بالبرنامج، حيث توجد أدوات وتقنيات تساعدنا في أداء هذه المهمة بطريقة سهلة مقارنة ببعض اللغات السابقة التي كانت تتطلب جهداً كبيراً من المبرمج لفحص البرنامج للتأكد من خلوه من الأخطاء.

ويرتكب أي مبرمج سواء كان خبيراً أو مبتدءاً أخطاء أثناء كتابة البرنامج، لهذا فإن فحص البرامج وتجربتها تعتبر جزءاً أساسياً من تصميم البرامج.

لكتابة برنامج خال من الأخطاء إلى حد ما نفذ الخطوات التالية:

١- جرب تنفيذ الأوامر **Testing** للتأكد من أنها تعمل بالشكل الصحيح.

٢- افحص البرنامج **Debugging** عن طريق تنفيذ سلسلة من التجارب للتأكد من أن البرنامج ككل يعمل بالشكل المطلوب، ففي الخطوة الأولي قد تقوم بتجربة جزء من البرنامج وخصوصاً إذا كان البرنامج يتكون من مجموعة كبيرة من الأجزاء، بينما يتم في هذه الخطوة فحص كل الأجزاء معاً للتأكد من عملها وترابطها بالشكل المطلوب. وتجربة الأمر للتأكد من عمله ولا يعنى هذا أن البرنامج سيعمل بالطريقة المطلوبة، لهذا لابد من إجراء سلسلة من عمليات الفحص للتأكد من أن البرنامج سيعمل كما هو مخطط له، ويمكن وصف دورة تصميم البرامج بالشكل التالي:



دراسة وفهم المشكلة التي سيكتب لها البرنامج.

١- تصميم وتخطيط البرنامج.

٢- كتابة البرنامج.

٣- فحص الأخطاء بالبرنامج.

٤- فحص عمل البرنامج.

وفي حالة وجود أخطاء بالمرحلتين الرابعة والخامسة يتم العودة إلى النقطة الثالثة والتي تتضمن إعادة كتابة البرنامج.

ولكتابة برنامج خال من الأخطاء تذكر ما يلي:

- يجب أن تفترض أن البرنامج لابد وأن به خطأ ما، فمهما كان المبرمج خبيراً لابد أن يقع في الأخطاء.
- حاول تجربة البرنامج باستخدام بعض البيانات الافتراضية، وهذه التجربة يجب تكرارها عدة مرات، لكي تتأكد من عمل البرنامج.

- حاول استخدام بيانات غير منطقية لمعرفة رد فعل البرنامج تجاهها، مثلاً في العمليات الحسابية لا يقبل الحاسب القسمة علي صفر، وتجربة العديد من الأرقام غير المنطقية قد يساعد علي اكتشاف الأخطاء أفضل مما لو استعملت أرقام مشابهة للأرقام الفعلية.
- عند الانتهاء من كل جزء من البرنامج، حاول تجربته علي حدة لتتبع الاخطاء بكل جزء أفضل مما لو جربت البرنامج بشكله الكامل.
- جزء البرنامج إلى أقسام كلما أمكن، لتتمكن من اكتشاف وتصحيح الأخطاء بطريقة أفضل.

أنواع الأخطاء في البرامج:

١- الأخطاء التركيبية.

٢- الأخطاء التنفيذية.

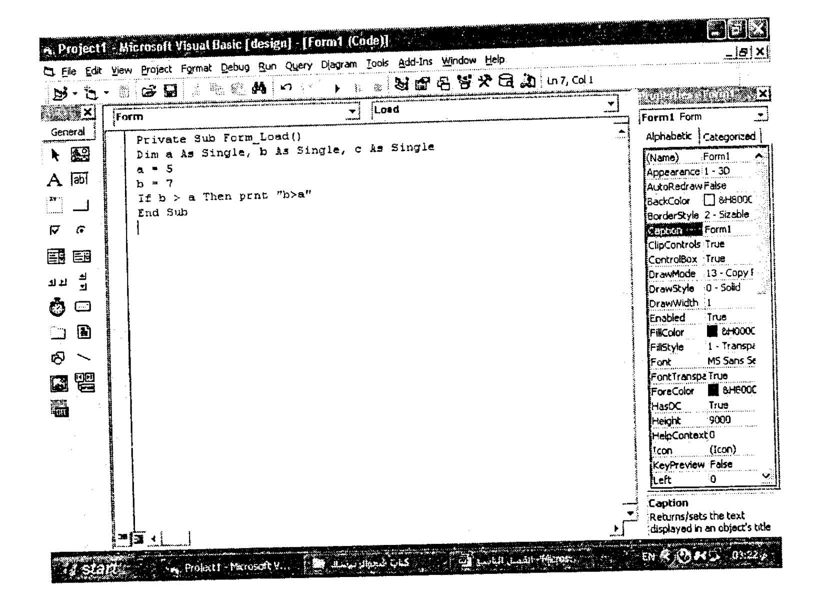
٣- الأخطاء المنطقية.

الأخطاء التركيبية:

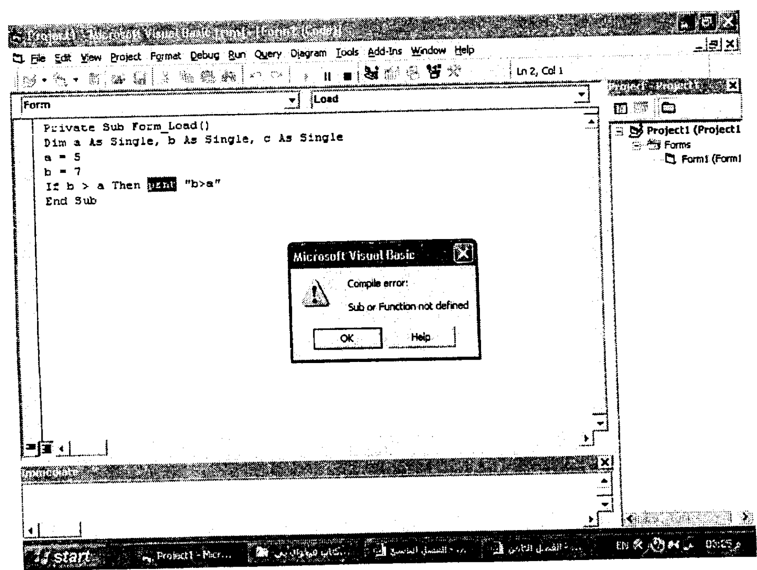
الأخطاء التركيبية، عبارة عن أمر أو أوامر لا تخضع للشروط الصحيحة لكتابة الأمر، مثل كتابة الأمر بشكل خطأ كما لو كتبت أمر **Print** كما يلي **Prent**. وقد يكون الخطأ عبارة عن استخدام إشارة غير صحيحة بالبيانات أو المعادلات، والأوامر التي لا تكتب طبقاً لقواعد كتابتها تعجز اللغة عن متابعة تنفيذها وبالتالي يتوقف البرنامج عن التنفيذ.

اكتب البرنامج التالي كنموذج لهذا النوع من الأخطاء:

```
Sub Form_Load ()
Dim a As Single, b as single, c As Single
A = 5
B = 7
If b > a Then Prent "b>a"
End sub
```



نفذ البرنامج وستجد رسالة وجود الخطأ كما يلي:

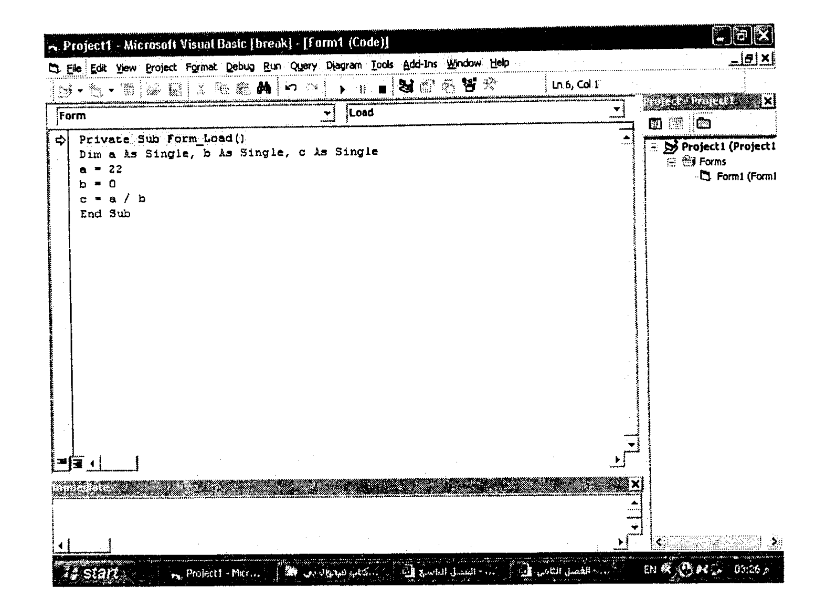


الأخطاء التنفيذية:

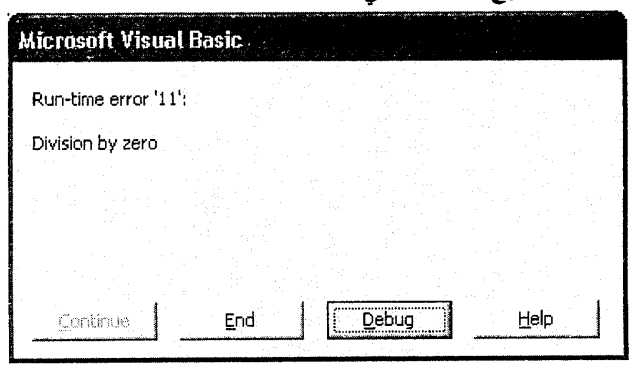
هذه الأخطاء ليس لها علاقة بأوامر البرنامج بقدر ما هي مرتبطة بالحاسب الـذي ينفـذ البرنامج، فمثلاً تنفيذ القسمة علي صفر تعتبر صحيحة كأمر بالبرنامج، ولكن الحاسـب يعجز عن القيام بالقسمة علي صفر، لأن الصفر غير محدد القيمة للحاسب. أيضـاً قـد تحدث الأخطاء التنفيذية عند استخدام وسيط للتخزين غير جاهز للعمل فيعجز البرنامج عن متابعة الأمر ويتوقف عن العمل.

اكتب البرنامج التالي كنموذج علي هذا النوع من الأخطاء:

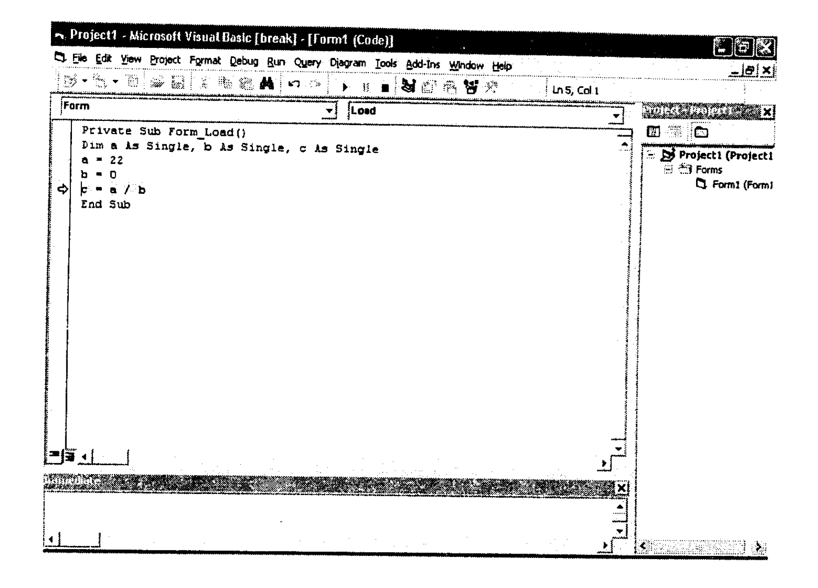
```
Sub Form_Load ( )
Dim a As Single, b as single, c As Single
A = 22
B = 0
C = a / b
End sub
```



ستظهر رسالة تحدد نوع الخطأ كما يلي:



وعند الضغط علي **DEBUG** يعرض مكان الخطأ كما يلي:



الأخطاء المنطقية:

يقع الخطأ المنطقي عند تنفيذ البرنامج من البداية إلى النهاية دون حدوث أي مشاكل، ولكن نكتشف أن النتائج التي حصلنا عليها لا تطابق تلك التي تم توقعتها من البرنامج، فمثلاً لو أردنا معرفة متوسط ٣ أرقام، فإذا كتبنا الأمر التالي = ٣٤ + ٧٨ + ٤٥ ÷ ٣. فإن هذا يعتبر خطأ منطقي وليس خطأ تركيبي أو تنفيذي، حيث يجب أن يكتب الأمر بالشكل التالي = (٣٤ + ٧٨ + ٤٥) / ٣.

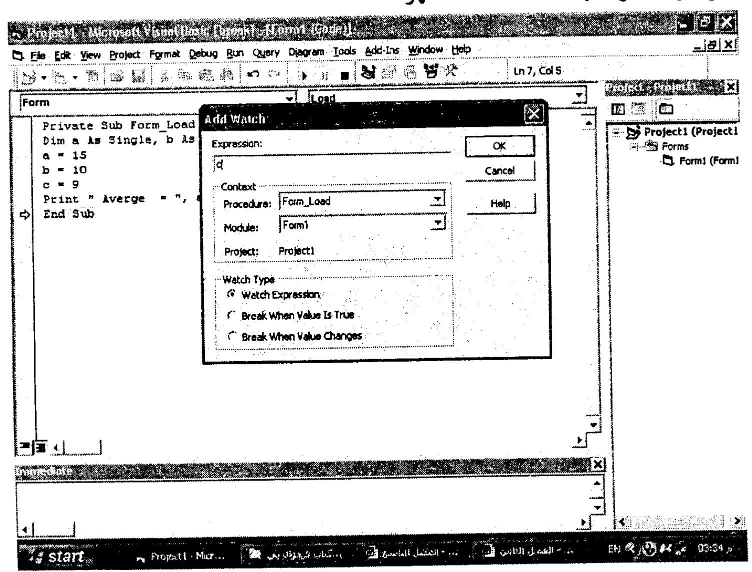
والأخطاء المنطقية لا يمكن اكتشافها بسهولة، حيث علي المبرمج تتبع مسار البيانات من لحظة دخولها بالبرنامج إلى مرحلة معالجتها إلى مرحلة النتائج، أي مراجعة البرنامج بالكامل.

ويمكن التغلب علي هذا النوع من المشاكل بتقسيم البرنامج إلى أجزاء، بحيث يكون كل جزء مسئول عن مهمة معينة، مما يساعد علي معرفة الخلل بأي جزء بالبرنامج. اكتب البرنامج التالي كنموذج علي هذا النوع من الأخطاء:

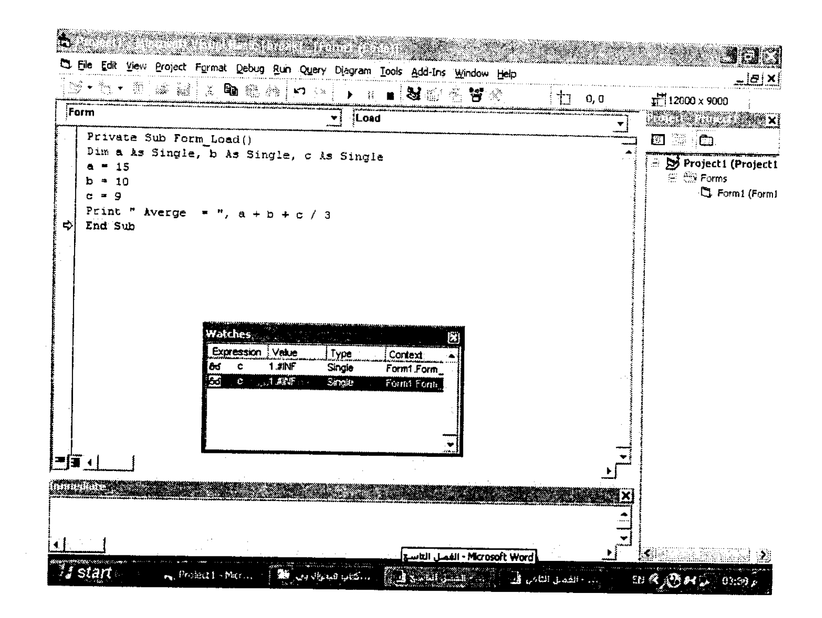
```
Sub Form_Load ( )
Dim a As Single, b as single, c As Single
A = 15
B = 10
C = 9
Print "Average = ", a+b+c/3
End sub
```

للتغلب علي الأخطاء التي يمكن أن تحدث، يزودنا برنامج فيجوال بيسك بعديد من الأدوات التي يمكن استخدامها للتعرف علي مشاكل البرنامج وحلها بشكل سريع، وستجد هذه الأدوات بقائمة التصحيح **Debug**. والتي تحتوى علي الأوامر التالية:

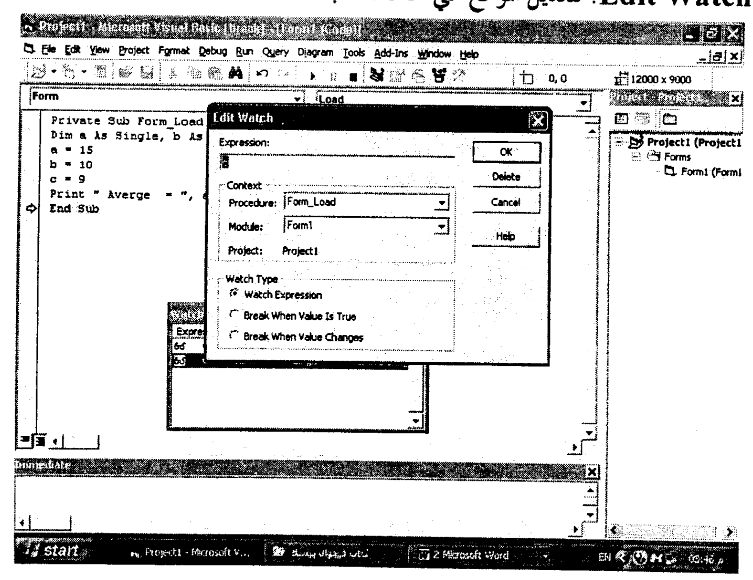
Add Watch: لمراقبة نتائج ومحتوى مواقع البيانات، ويمكنك تحديد عدة مواقع في نفس الوقت لمراقبتها بشاشة خاصة تظهر لك.



Instant Watch: تعمل هذه الإمكانية مثل السابقة ولكنها تنطبق علي موقع واحد فقط يحدد مباشرة بالبرنامج.



Edit Watch: لتعديل المواقع التي حددت سابقاً.



Calls: قد يتكون البرنامج من عدة أجزاء وبالتالي تتوزع البيانات علي عدة أجـــزاء، هذه الإمكانية تقوم بتتبع كل موقع حدد، بحيث تظهر لنا هذه المواقع والأجـــزاء الـــتي استخدمت ضمنها.

Single Step: لتنفيذ أحد الأوامر علي حدة لمراقبة هذه الخطوة بالبرنامج.

Procedure Step: الإجراء أو الوظيفة مجموعة من الأوامر، ولكن يمكن تنفيـــذها وكأنها أمر واحد، بدلاً من تتبع كل الأوامر بالأجراء والوظيفة.

Toggle Breakpoint: لإضافة علامات توقف بالبرنامج، عند تنفيذ البرنامج ليتم إنجاز الخطوات التي تسبق المكان الذي وضعت به العلامة.

Clear all Breakpoint: لمسح علامات التوقف من البرنامج، حيث يمكن وضع عدة علامات بالبرنامج.

Set Next Statement: لتحويل التنفيذ من الخطوة الحالية إلى خطـــوة أخـــري بالبرنامج، مما يتيح لك تجاهل أوامر تسبق الأمر الذي سينفذ لاحقاً.

Show Next Statement: أثناء قطع البرنامج عن العمل، يتم تلقائيـــاً حفـــظ الخطوة التي يحل تنفيذها، قد تقوم بنقل المؤشر بين أجزاء البرنامج، ولكي تعرف ما هـــي الخطوة التالية التي سيتابع منه البرنامج العمل اختار هذه الإمكانية.

إيقاف البرنامج عن العمل:

إذا عمل البرنامج الذي كتبته بطريقة لا يمكن التحكم فيها، يمكن قطع عمل البرنـــامج باستخدام المفاتيح **Ctrl + Break** معاً، وقد لا يتوقف البرنامج عن العمل في بعض الأحيان لأن بعض الشروط أو الروتين المستخدمة قد تدور في حلقة مفرغة ولهذا لن يصل

البرنامج إلى نهاية، وبالتالي سيدور في حلقة مفرغة ما لم تقم بإيقافه بالمفـــاتيح المـــذكورة أعلاه.

ويمكن متابعة عمل البرنامج باستخدام قائمة **Run** ثم اختار **Restart** أو اســـتخدام مفتاح **F5**.

ملحوظة:
عند استخدام مفاتيح التوقف ستجد أن الأمر الذي وصل إليه البرنامج قبل التوقف قـــد أحيط بمستطيل للتعرف علي أن التنفيذ وصل إلى هذا المكان.

وضع علامة توقف:
تساعد علامة التوقف علي تنفيذ الأوامر من جزء إلى آخر لمراقبة تنفيذ هـــذه الأوامـــر. ولكي تضع علامة التوقف بالبرنامج نفذ الخطوات التالية:

- اظهر الأوامر علي شاشة كتابة الأوامر.
- اختار الأمر المطلوب.
- اختار قائمة **Debug**.
- اختار **Add Breakpoint** أو استخدم مفتاح **F9** مباشرة.
- يمكن إزالة علامة التوقف باستخدام مفتاح **F9** مرة ثانية.

بعد وضع العلامة نفذ البرنامج وستري بعد ذلك أن البرنامج قد توقـــف عـــن العمـــل وظهرت لك أوامر البرنامج علي الشاشة، ويمكنك متابعة التنفيذ باستخدام **F5**.

مراقبة محتويات مواقع البيانات:
يمكن مراقبة محتويات مواقع البيانات أثناء تنفيذ البرنامج للتأكد من أن هذه المواقع تحوى البيانات المطلوبة، ولكي تراقب هذه المواقع نفذ الخطوات التالية:

- اعرض أوامر البرنامج.

- حدد اسم الموقع بالفأرة.
- اختار قائمة **Debug** ثم اختار **Add Watch**.
- كرر العمل بالنسبة للمواقع الاخري إن وجدت.

بعد تنفيذ الخطوات السابقة شغل البرنامج لكي تراقب محتويات المواقع.

ملحوظة:

يجب وضع علامات بالبرنامج لكي يتاح لك مراقبة بيانات المواقع خطوة بخطوة.

تنفيذ البرنامج خطوة بخطوة:

من الوسائل التي يمكنك الاستفادة منها للتحقق من عمل البرنامج، نجد استخدام ميـــزة التنفيذ خطوة بخطوة، نفذ الخطوات التالية:

- اكتب أوامر البرنامج.
- ضع الفأرة في المكان الذي سيستخدم لتنفيذ البرنامج.
- اضغط **F8** لتنفيذ أمر واحد.
- تابع الضغط علي مفتاح **F8** إلى أن تصل إلى نماية البرنامج.

الأخطاء الخارجية:

يحتاج تتبع الأخطاء إلى صبر، لأن الأخطاء يمكن أن تحدث لأسباب متعـــددة، وتعتمـــد المتابعة علي مراحل تصميم واستخدام البرنامج، فالأخطاء التي تقع بالبرنامج قد تكـــون أخطاء في كتابة البرنامج، ولكن إذا تأكدنا من أن البرنامج يعمل بطريقة صـــحيحة ولا يوجد به أخطاء هل يعنى هذا أن البرنامج سيعمل بالطريقة المطلوبة، الخطأ وارد، ذلك أن الخطأ الذي يحدث الآن ليس له علاقة بأوامر البرنامج أو تخطيطه، بل قد تقـــع أخطـــاء خارجية ليس للبرنامج علاقة بها، ولكن وجود هذه الأخطاء يؤدى إلى تعطيل البرنامج عن العمل المطلوب منه.

فمثلاً، نفترض أن البرنامج به أمر طباعة بيانات علي الطابعة، وفي لحظة تنفيذ الأمر كانت الطابعة غير جاهزة أو معطلة. سيفشل هذا الأمر في التنفيذ ويتوقف البرنامج عن العمل، وللتغلب علي هذا النوع من الأخطاء، توجد قائمة بمجموعة من الأخطاء الخارجية المحتملة أثناء تنفيذ البرنامج، ويتم من خلال هذه المجموعة تخمين الخطأ مسبقاً والعمل علي إصلاحه من خلال البرنامج وبالتالي لن يتوقف البرنامج عن العمل إذا طرأ خطأ خارجي أثناء تنفيذه، لأننا نكون متوقعين مسبقاً حدوث هذا الخطأ وحددنا طريقة إصلاحه عند حدوثه.

وبلغة فيجوال بيسك مجموعة من الأخطاء المحتمل وقوعها أثناء تنفيذ البرنامج، ولكل منها رقم خاص به، ويستطيع المبرمج استخدام هذا الرقم لمعالجة الخطأ، حيث يتعرف علي الخطأ بالرقم الخاص به، وبهذا يستطيع المبرمج تخمين الخطأ وكتابة الأوامر التي تساعد علي تخطيه.

أنواع الأخطاء الخارجية أثناء تنفيذ البرنامج:

١- الأخطاء العامة:

وأرقامها من ٣ إلي ٥٢١ وهى تتعلق بالملفات والفهارس والطابعة وأنواع البيانات بشكل عام.

٢- الأخطاء الخاصة بتبادل البيانات:

وأرقامها من ٣١٠٠١ إلي ٣٢٧٦٥ يحدث هذا النوع من الأخطاء أثناء استخدام أوامر البيانات بين تطبيقات برنامج ويندوز، مثلاً تستطيع كتابة برنامج ليقرأ جدول من برنامج أكسل ونقله إلي برنامج وورد باستخدام أوامر فيجوال بيسك.

٣- الأخطاء الخاصة بالشاشة المعروضة:

وأرقامها من ٣٠٠٠٠ إلي ٣٠٠١٩، هذا النوع من الأخطاء خاص بتحديد أبعاد ووضع الصور بالشاشة.

٤- الأخطاء الخاصة بقواعد البيانات الخارجية:

وأرقامها من ٦٠٠ إلي ٦٤٨، تقع هذه الأخطاء أثناء قراءة ملفات قواعد البيانات التي كونت باستخدام **dbase** أو **Foxpro**.

٥- الأخطاء الخاصة بملفات **Access**:

وأرقامها من ٣٠٠١ إلي ٣٢٩٩، وتقع هذه الأخطاء أثناء استخدام ملفات برنامج اكسس، فالارتباط بين فيجوال بيسك وبرنامج اكسس قوى جداً، لأن شركة ميكروسوفت تحاول تصميم لغة موحدة لكل تطبيقات برامج ميكروسوفت دون أن يكون هنالك لغة خاصة بكل برنامج مما يساعد المبرمج علي تطوير قدراته، لأنه يستطيع استخدام لغة غنية ومتعددة الجوانب.

مراجعة الأخطاء:

الأخطاء التي ذكرنا أرقامها يمكن مراجعتها من خلال المرجع الخاص باللغة، وستجدها ببرنامج المساعدة، وللوصول إلى هذه الأخطاء لمعرفة كافة التفاصيل عنها نفذ الخطوات التالية:

١- اختار **Help**.
٢- اختار **Content**.
٣- اختار **Programming Language**.
٤- اختار من الشاشة التي تظهر لك حرف "**O**".
٥- اختار **On Error..**.
٦- اختار **See also**.
٧- اختار **Trappable Errors**.

ستظهر لك مجموعة من الأخطاء مع أرقامها، وإذا اخترت أحد هذه الأخطاء سيظهر لك كيفية استخدام الأمر المناسب لحل الخطأ المتوقع حدوثه بالبرنامج.

أوامر معالجة الأخطاء الخارجية:

يمكن معالجة الأخطاء الخارجية بالبرنامج عند وقوعها، ولتصحيح الخطأ علينا اســتخدام أوامر مخصصة لذلك وهي:

١- On error Goto: لتوجيه البرنامج للانتقال إلى مكان آخر بالبرنامج، يخصص لمتابعة الأخطاء ومعرفة نوع الخطأ وبالتالي إعطاء التوجيهات اللازمة لتخطى الخطأ المحتمل.

٢- Err: لمعرفة رقم الخطأ الذي حدث، ويتم استخدام هذا الرقم من قبــل المــبرمج لمعرفة رقم الخطأ الذي حدث وتحديد الخطوة التالية بناء علي رقم هذا الخطأ، لأن لكــل خطأ معالجة تختلف عن الأخطاء الاخري، لهذا فإن هذا الرقم هام في تحديد الخطأ الــذي حدث والخطوات التي يجب اتخاذها للتغلب عليه.

٣- Errl: لعرض رقم السطر الذي به الأمر الذي حدث به الخطأ، لغة فيجوال بيسك تسمح باستخدام أرقام لسطور البرنامج بالأسلوب المتبع في كتابة البرنامج بلغة بيســك، ولهذا فإن هذا الأمر يعمل إذا تم كتابة أرقام الخطوات، أمــا إذا لم تســتخدم الأرقــام فسيعرض قيمة صفر لأنك لم تستخدم أرقام لسطور للبرنامج.

٤- Error: لعرض اسم الخطأ، مثلاً رقم الخطأ الذي يحدث أثناء محاولة الطباعة هــو ٥٧، اسم هذا الخطأ هو device 1/0، وبالتالي فإن أمر Error سيعرض لك اســم الخطأ.

٥- Resume: لإعادة تنفيذ الأمر الذي تسبب بالخطأ، مثلاً إذا أعطيت أمر للطابعة وأثناء تنفيذ هذا الأمر لم تكن الطابعة جاهزة لسبب ما، في هذه الحالة يتم توجيه البرنامج لينتقل إلى الجزء الخاص بالتحقق من الأخطاء المتوقعة، ويتم متابعة الخطأ من خلال رقــم الخطأ، وبالتالي يتم اتخاذ الخطوات لإصلاح موقف الطابعة من مستخدم البرنامج، وبعــد ذلك علينا إعادة تنفيذ الأمر الذي لم يتم تنفيذه نظراً للخطأ الذي حدث، لهذا فإن أمــر Resume يعيد تكرار تنفيذ الأمر الذي لم يكتمل تنفيذه بسبب الخطأ الذي حدث.

```
Sub Form_Click ( )
1000 Dim A, B, C
1010 on Error Goto ErrHandler
1020 B = 1
1030 A = BIc
1040 Exit Sub
```

```
ErrHandler:
MsgBox "Error number" & Err &" Occurred at line "
& Erl
Resume Next
End Sub

Sub Form_Load ()
End Sub
```

نماذج لاستخدام الأوامر المتعلقة بالأخطاء الخارجية:

١- يقوم البرنامج التالي بفتح ملف من الاسطوانة، والافتراضات التي يمكن أن تحـــدث أثناء محاولة فتح الملف هي:

- أن الملف غير موجود.
- أن الفهرس غير موجود.
- أن وحدة الاسطوانات أو الاسطوانة غير جاهزة.

بناء علي هذه الافتراضات يتم اتخاذ الخطوات اللازمة لمعالجة الخطأ الذي حدث.

ولكتابة هذا البرنامج نفذ الخطوات التالية:

- بواجهة البرنـــامج **Form1** كـــون٢ أزرار الأوامـــر **command Button**، غير الاسم إلى فتح ملف البيانات ثم الاسم الثاني إلى إنهاء.
- اضغط بالفارة علي الزرار الأول فتح ملف البيانات ثم اكتب الأوامر التالية.
- اضغط بالفأرة علي زرار إنهاء ثم اكتب أمر **End**.

٢- هذا البرنامج يعرض نوع الخطأ ورقم السطر الذي تم به الخطأ، لكتابـــة البرنـــامج اضغط علي واجهة البرنامج ثم اختار **Click** من الصندوق المقابـــل للعنـــوان **Proc** فالبرنامج سيعمل عندما تضغط علي واجهة البرنامج فقط.

```
Sub Command1_Click ()
```

```
Sub form_Click ( )
Dim Drive, Msg
On Error Go To Errorsection
Msg = "حاولت فتح ملف غير موجود"
Msg = Msg & "علي الاسطوانة لذلك فشل الأمر"
Msg = Msg & "سيعرض قسم اكتشاف الأخطاء رسالة"
Msg = Msg & "توضح الخطأ الذي حدث"
MsgBox Msg
Open " B/ TEST / X.DAT " For Input As #1

Close # 1
Exit Sub

Errorsection:
Select Case Err
Case 53 : Msg = ":الملف غير موجود الخطأ رقم ٥٣ "
Case 68 : Msg = " : وحدة الاسطوانات" غير متاحة خطأ رقم ٦٨"
Case 76 : Msg = ". هذا المسار غير موجود خطأ رقم ٧٦ "
Case Else: Msg = "الخطأ رقم"& Err & " .حدث "
End Select
MsgBox Msg
Resume Next
End Sub

End Sub

Sub Command2_Click ( )
End
End Sub
```

الفصل الثالث عشر

التعامل مع الملفات

إنشاء ملفات البيانات

يتم تشغيل ملفات البيانات بلغة فيجوال بيسك باستخدام الملفات المتتالية **Sequential Access Files** أو الملفات المباشرة **Direct Access Files** أو الملفات الثنائية **Binary Files** في حفظ البيانات والوصول إليها، وتختلف الأوامر بناء علي طريقة أنشاء الملف، والطريقة المثلي في التعامل مع ملفات البيانات تتم باستخدام الملفات المباشرة في حفظ البيانات والوصول إليها، وذلك للأسباب التالية:

- **إمكانية قراءة / كتابة البيانات في نفس الوقت.**
- **الوصول المباشر: يمكن الوصول إلى البيانات بالملف بطريقة مباشرة وذلك بتخطي البيانات الاخري.**
- **تعديل السجلات: من السهولة تعديل السجلات، لأن الملف المباشر يسمح بفتح الملف للقراءة والكتابة في نفس الوقت مما يمكن من تعديل السجل مباشرة.**

ولتوضيح سرعة استخدام الملفات المباشرة نجد أن الملف المتتالي يقرأ كل البيانات السابقة علي البيان المطلوب، وكلما كانت البيانات كثيرة فإن استخدام الملف المباشر يساعد في الوصول إلى البيانات في زمن أقل من استخدام الملف المتتالي.

ويساعد استخدام الملفات المباشرة في الوصول إلي البيانات بطريقة مباشرة، ولا يختلف وقت الوصول إلى البيانات مهما كان موقعها، أيضاً يمكن من قراءة البيانات أو حفظها في نفس الوقت. ومعظم برامج قواعد البيانات تستخدم أسلوب الملف المباشر في معالجة البيانات.

ولكل أسلوب مزايا وسلبيات، ومن سلبيات الملفات المباشرة.

التركيب المحدد للسجل:

يجب أن تكون كل السجلات بالملف متساوية في الطول، مما يؤدى إلى وجود مساحات فارغة غير مستغلة، ذلك أن البيانات تختلف من سجل إلي آخر.

التوافقية:

تركيب البيانات بالملف المباشر غير متوافقة مع البرامج الاخري مثل اكسس أو اكسل أو وورد، بينما الملف المتتالي متوافق مع هذه البرامج.

الأوامر:

ستحتاج إلى عدد اكبر من الأوامر لمعالجة الملفات المباشرة مقارنة بالملفات المتتالية.

تصميم الملف:

يتكون الملف المباشر من مجموعة من السجلات، ويتكون السجل من مجموعة من الحقول، لذلك يجب:

- تحديد اسم كل حقل.
- تحديد طول كل حقل.
- تحديد نوع كل الحقل.

لحفظ ملف للموظفين، علينا تحديد اسم كل حقل من السجل وطوله ونوعه كما يلي:

١- اسم الموظف – حرفي – الطول ٢٥ حرف.
٢- المهنة – حرفي – الطول ٢٠ حرف.
٣- المرتب – رقمي – الطول ٥ خانات رقمية.
٤- رقم التليفون – حرفي – الطول ١٥ حرف.

٥- العنوان – حرفي – الطول ٤٠ حرف.

وبناء علي عدد الحروف الخاصة بكل حقل يتم تحديد طول السجل، والذي يساوى في هذه الحالة مجموع طول الحقول، طول السجل فى المثال السابق ١٠٥ حرف.

أوامر الملفات المباشرة:

لكل نوع من الملفات بلغة فيجوال بيسك أوامر تختلف عن أوامر الأنـــواع الاخـــري، وأوامر الملف المباشر هي:

Open: لفتح ملف البيانات.

Close: لغلق ملف البيانات.

Put: لحفظ السجل علي الاسطوانة.

Get: لقراءة السجل من الاسطوانة إلى الملف.

Eof: للتحقق من نهاية الملف.

Lof: لمعرفة حجم الملف.

Seek للانتقال إلى سجل معين.

Len: لمعرفة عدد الحروف المقروءة بالسجل الحالي.

Loc: لنقل مؤشر قراءة السجل إلى نهاية البيانات بالشبكة للسماح بمشـــاركة عـــدة أشخاص بالملف.

Lock Read: أمر لمنع الأشخاص العاملين علي الشبكة من قراءة الملف.

Lock Write: أمر يسمح للأشخاص العاملين علي الشبكة بقراءة الملف فقط.

Lock Read Write: أمر يلغي السماح للأشخاص العاملين علي الشبكة مـــن استخدام الملف.

Read: لتحديد الملف للقراءة فقط أثناء عملية الفتح.

Write: لتحديد الملف للكتابة فقط أثناء عملية الفتح.

Read Write: أمر لتحديد الملف للقراءة والكتابة معاً.

تدريب:

لاستخدام الملف المباشر اكتب ما يلي لتصميم واجهة البرنـــامج وكتابتـــه ثم إضـــافة البيانات:

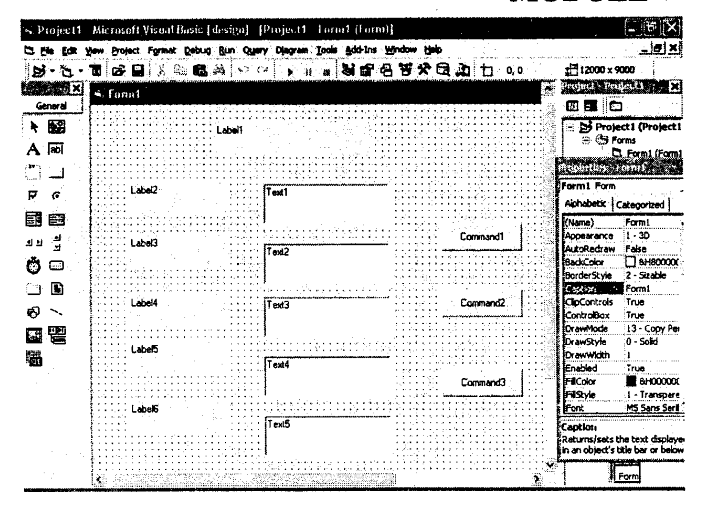
١- صمم الواجهة كما يلي:

Command Button عدد ٣

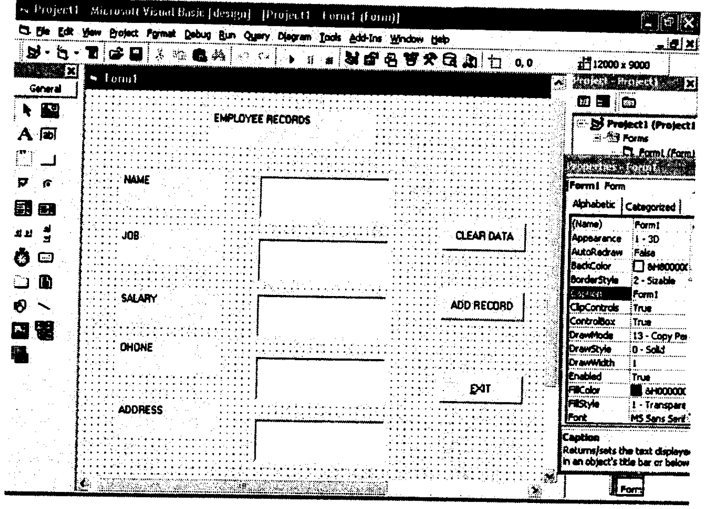
LABEL عدد ٦

Text عدد ٥

MODULE عدد ١



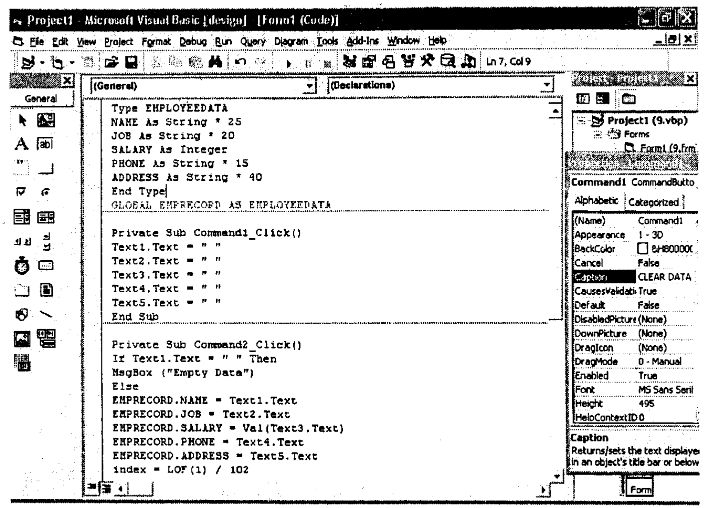
٢- عدل مواصفات الأشكال كما يلي:



بتغيير محتوى الخاصية **CAPTION**.

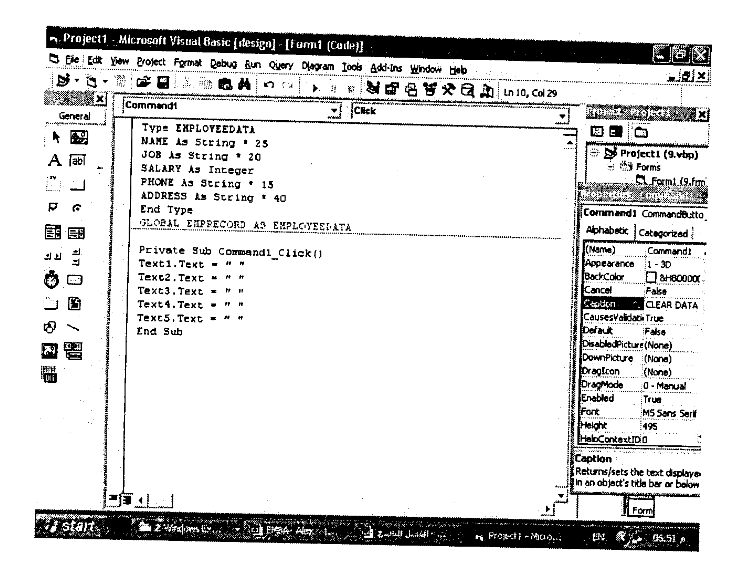
٣- أنشئ **Module** من قائمة **Project** ثم اكتب الأوامر التالية:

```
Type employeedata
Name As string *25
Job As string *20
Salary As Integer
Phon As String *15
Address As String *40
End Type
Global emRecord As employeedata
```



٤- اكتب الأوامر التالية لكل كائن وذلك باختيار الكائن ثم الضغط عليه مرتين فيتم فتح شاشة كتابة أوامر الكائن الذي تم اختياره، وعند الانتهاء اضغط علي **ALT + F4** كما يلي:

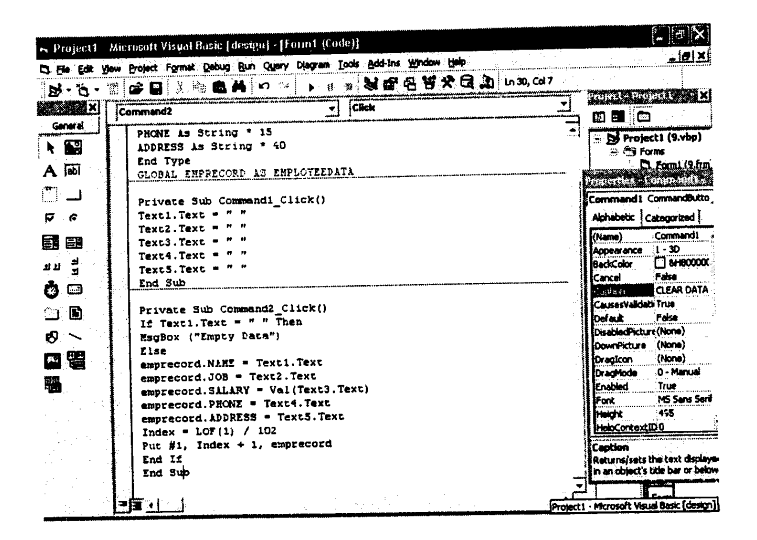
```
Sub Command 1_Click ()
Text1.Text = " "
Text2.Text = " "
Text3.Text = " "
Text4.Text = " "
Text5.Text = " "
End Sub
```



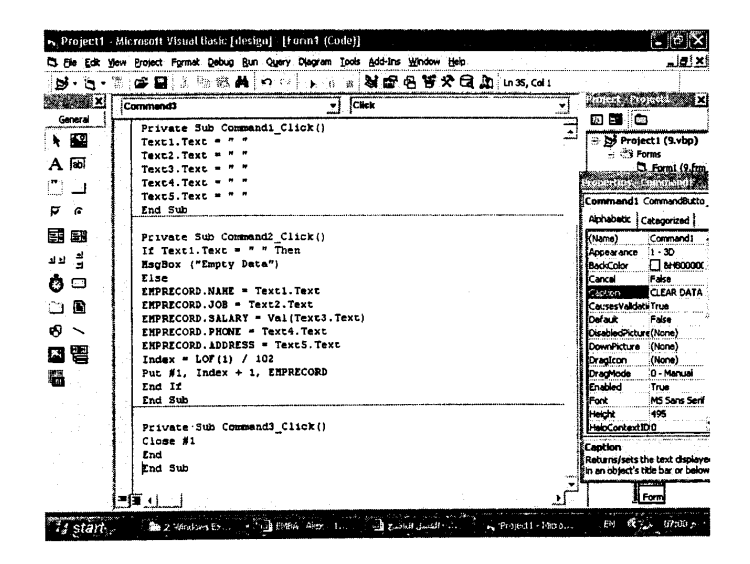

```
Sub Command2_Click ()

If  Text1.Text = " " Then
MsgBox ("Empty Data ")

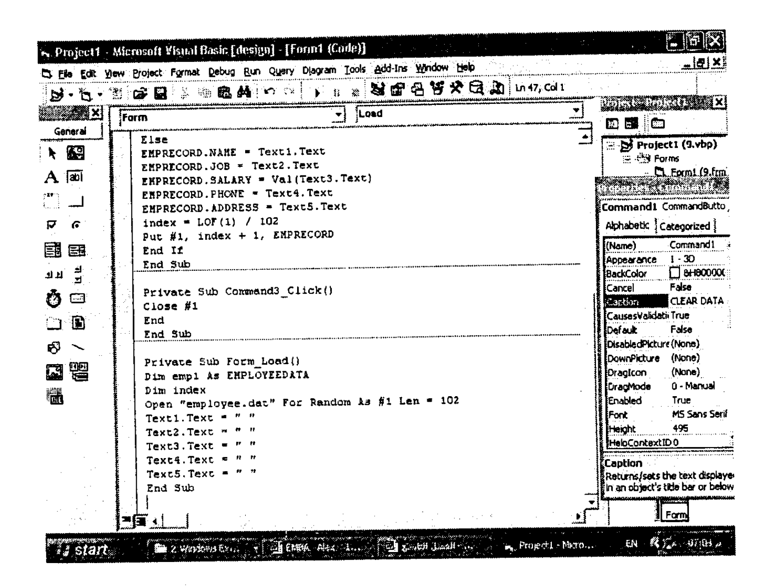
Else
Emprecord.nam = Text1.Text
Emprecord.job = Text2.Text
Emprecord.salary = Val (Text3.Text)
Emprecord.phon = Text4.Text
Emprecord.Address = Text5.Text
Index = Lof (1)/102
Put #1, index + 1, emprecord
End If
End Sub
```




```
Sub Command3_Click ()
Close # 1
End
End Sub
```



Dim emp1 Record As employeedata
Dim index
Open "employee.dat" For Randon As #1 Len= 102
Text1.Text = """
Text2.Text = """
Text3.Text = """
Text4.Text = """
Text5.Text = """
End Sub



٥- نفذ البرنامج وادخل البيانات به.

الفصل الرابع عشر

التعامل مع الملفات

تعديل السجلات

تتعرض البيانات التي نتداولها حول موضوع معين للتغيير ولا يمكن أن تكون ثابتة بشكل دائم، فعنوان شخص معين يتغير إذا انتقل إلى مكان آخر، لهذا بيانات العنوان القديم يجب استبدالها ببيانات العنوان الجديد. وعند حفظ البيانات باستخدام الحاسب يجب أن نتمكن من تحديث البيانات المخزنة لتطابق بيانات الواقع الحالي، درسنا كيفية حفظ البيانات وكيفية قراءتها، وسندرس كيفية تعديل أحد السجلات بقراءة السجل المطلوب ثم تحديث البيانات السابقة بالبيانات الجديدة.

والبرنامج التالي يشبه عمل برنامج القراءة وبرنامج التسجيل الذين قدماً سابقاً، ولهذا سنعيد استخدام الأوامر السابقة، ويمكن إعادة استخدام الواجهة الخاصة ببرنامج القراءة مع إدخال بعض التعديلات عليها لتلائم عمل البرنامج الجديد، ويفضل إعادة تكوين الواجهة وكتابة البرنامج لنتمرن علي طريقة كتابة البرامج بلغة فيجوال بيسك لصقل هذه المهارة.

تصميم واجهة البرنامج:

واجهة البرنامج التي نستخدمها للتعامل مع البرامج تتكون من العناصر التالية:

- ٧ أشكال من نوع **Label** مخصصة لكتابة العناوين التالية

Employee Record
Name
Job
Salary
Phone
Address
Scroll up Down

ويمكن كتابة هذه البيانات باختيار كل شكل ثم تعديل مواصفاته بالضغط علي مفتاح **F4** ثم تغيير صفة **Caption** لتلائم العناوين المطلوبة بدلاً من ظهور اسم **Label** بكـــل شكل.

- ٥ أشكال من نوع **Text**، تخصص لعرض بيانات الســـجل واســـتقبال البيانات الجديدة.
- ٢ زرار من نوع **Command** تخصص لإنهاء البرنامج وتعديل البيانات، ويتم تغيير الرسالة التي تظهر علي كل زرار بالضغط علي مفتاح **F4**.
- عمود تحريك راسي **Vertical Scroll Bar**، يستخدم هذا العمود لقراءة السجلات من الملف وعرضها علي الشاشة الخاصة بذلك.

أوامر البرنامج:

يتم كتابة البرنامج باستخدام واجهة البرنامج **Form** وباستخدام برنـــامج **Module** مخصص لتعريف مواقع التخزين العامة، ويستخدم لكتابة الإجراءات والوظـــائف. لهـــذا سنبدأ بتكوين **Module** جديد باتباع الخطوات التالية:

- اختار **Project**.
- اختار **New Module**.
- ثم اكتب الأوامر التالية.

```
Type employeedata
Name As string * 25
Job As string * 20
Salary As Integer
Phon As String * 15
Address As String * 40
End Type
Global empRecord As employeedata
Global rec_no As Integer
```

الجديد في هذه الأوامر هي تعريف الموقع **Rec_no**، حيث سيستخدم للاحتفاظ برقم كل سجل تقوم بقراءته، وإذا عدلت أحد السجلات واخترت زرار التعديل فإن البرنامج سيستخدم هذا الزرار لكتابة البيانات الجديدة الخاصة بالسجل بدلاً من البيانات السابقة.

- بعد كتابة الموديول **Module** انتقل إلى واجهـــة البرنـــامج **Form** ثم اضغط الفارة مرتين متتاليتين بمكان خالي من الواجهة، واكتب الأوامر التي توجد بالجزء المسمى **Sub Load_Form**، بعد كتابة الأمر اضـــغط علي **Alt + F4** للعودة إلى الواجهة.

```
Sub Form_Load ( )
Dim emp1 Record as employeedata
Dim index
Open "employee.dat" For Random As #1Len = 102
Index = 1
Vscroll1.Min = 1
```

```
Vscroll1.Max = Lof 102
Text1.Text = " "
Text2.Text = " "
Text3.Text = " "
Text4.Text = " "
Text5.Text = " "
End Sub
```

* ضع الفارة علي زرار Update Record ثم اكتب الأوامر كما هي مبينة بالإجراء Sub Command1_Click ()، هذه الأوامر خاصة لإبدال بيانات السجل الحالي بالبيانات الجديدة، ويتم هذا باستخدام أمر # Put، والتسجيل مكان السجل السابق يعتمد علي معرفة رقم السجل المطلوب تعديله، وقد تمت معرفة رقم السجل المطلوب باستخدام الموقع المعنون Rec_no، بعد كتابة الأوامر استخدم مفتاحي Alt + F4 للانتقال إلى واجهة البرنامج Form.

```
Sub Command1_Click ()
Emprecord.name = Text1.Text
Emprecord.job = Text2.Text
Emprecord.salary = VAL (Text3.Text)
Emprecord.phon = Text4.Text
Emprecord.address = Text5.Text
Put #1, rec_no, emprecord
End Sub
```

* اضغط بالفارة علي Exit ثم اكتب أوامر إنهاء عمل البرنامج في الإجراء Sub Command2_Click ()، بعد كتابة الأوامر اضغط علي مفتاحي Alt + f4 للعودة إلى واجهة البرنامج.

```
Sub Command2_Click ()
Close # 1
End
End Sub
```

* اضغط بالفارة بشكل متتالي علي عمود التحريك الرأسي، واكتب الأوامر الخاصة بحفظ السجلات كما هي مبينة بالجزء المعنــون Sub Change_Scroll ()، التغيير الوحيد في هذا الإجراء عن البرنامج السابق هو استخدام المتغير Rec_no للاحتفاظ برقم السجل الذي نقرأه حالياً، فــإذا ضــغطت علــي زرار التعــديل Update Record، فإن الإجراء الخاص بهذا الزرار سيقوم بحفــظ البيانــات الجديدة بدلاً من البيانات القديمة بالسجل، ويتم هذا بمعرفة الرقم الخاص بالســجل الذي نرغب بتعديله، بعد كتابة الأوامر اضغط علي مفتاحي Alt + F4 للانتقال إلى واجهة البرنامج.

```
Sub Vscroll1_Change ()
Index = vscroll1.value
Get #1, index, emprecord
Text1.Text = emprecord.name
Text2.Text = emprecord.job
Text3.Text = emprecord.salary
Text4.Text = emprecord.phon
Text5.Text = emprecord.address
Rec_no = index
End Sub
```

تعديل البيانات:

لتعديل السجلات السابقة، شغل البرنامج بالطريقة العادية، ثم اضغط علي الأســهم التي تجدها بالعامود الراسي، وعندما تجد السجل الذي ترغب في تعديلــه، أضــف البيانات الجديدة ثم اضغط زرار Update Record، سيتم حفظ التعديل، ولو أعدت قراءة السجل مرة أخري باستخدام عامود التحريك الرأســي فســتري أن البيانات الجديدة قد حلت محل البيانات القديمة.

الفصل الخامس عشر

طرق التعامل مع قواعد البيانات في فيجوال بيسك

توجد أربعة طرق للتعامل مع قواعد البيانات وهي:

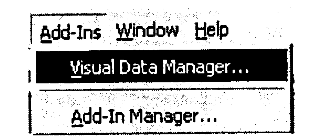
١	DATA	و هي إحدى المكونات الموجودة ضمنا في فيجوال بيسك تجدها في صندوق الأدوات وتعتبر أداة سهلة لربط قواعد البيانات البسيطة وتؤدي هذه الأداة بعض الإجراءات مثل الحذف والإضافة والتحرير والتحديث.
٢	كتابة الكود	الربط بالكود طريقة يجب على كل مبرمج قواعد بيانات أن يتعامل معها ويتقنها لكي يفتح أمامه آفاق برمجة قواعد البيانات وستجد أنها ليست بالصعوبة كما تتوقع من اسمها فهي عبارة عن أوامر معينة ثابتة تقريبا إذا استطعت أن تتأمل معها في مثال واحد ستتعامل معها في جميع ما يواجهك من أمثلة
٣	ADO	و هي تقنية جديدة من مايكروسوفت وعندما تتعامل معها ستجد تشابها بينها وبين الطريقة الأولى، ولكن هذه الطريقة تعطيك مدى أوسع وخيارات أفضل وستستمتع بهذه الطريقة عندما تتعامل مع منشيء بيئة البيانات الذي يعطيك تحكم أكبر في قواعد البيانات وبسهولة كبيرة
٤	Data Environment Design	منشيء بيئة البيانات وهي تقنية جديدة أيضا في فيجوال بيسك ٦ وتتعلق كثيرا بالتقنية السابقة

أولا: استخدام كائن Data:

هذه أولى طرق ربط قواعد البيانات، لربط قاعدة بيانات مع برنامجك في فيجوال بيسك ستحتاج أولا إلى وجود قاعدة بيانات لتربطها ببرنامجك ويتيح لك فيجوال بيسك أن تربط مع أنواع كثير من قواعد البيانات مرورا بأكسس ولوتس وإنتهاءا بفوكس برو وأوراكل، كما يوفر لك فيجوال بيسك عمل قاعدة بيانات بواسطة برنامج ملحق معه وهو غالبا يفي بحاجتك سننشأ قاعدة البيانات بهذا البرنامج المرفق مع فيجوال بيسك تحتوي جدول يحتوي على أسماء الطلاب وأرقامهم

الخطوة الأولى فتح فيجوال بيسك ثم الذهاب إلى قائمة

Add-ins >> visual data manager...

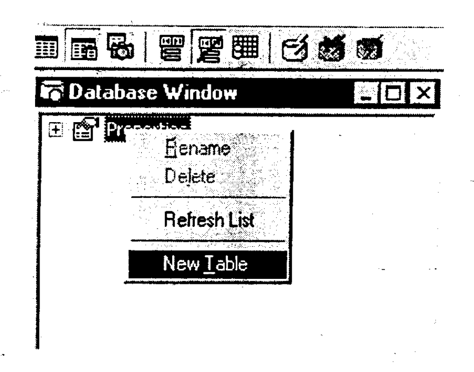


بعد ذلك ستفتح لك نافذة البرنامج أذهب إلى

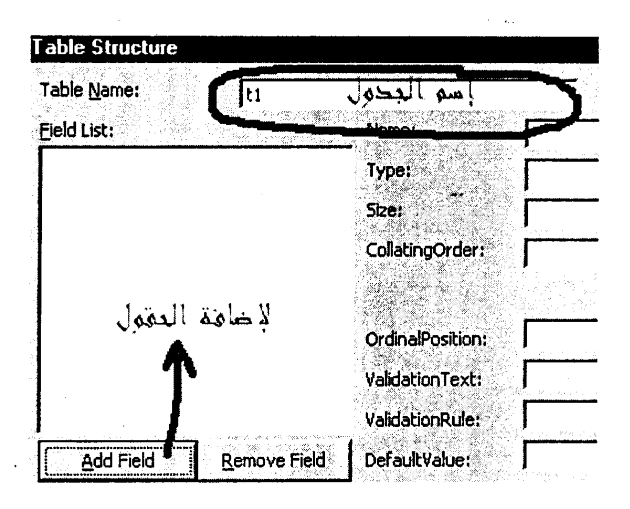
File>>new>>microsoft access>>version 7.0 mdb..

سيظهر لك صندوق حفظ حدد الموقع الذي تريد أن تحفظ فيه قاعدة بياناتك ثم أختر حفظ

بعد ذلك ستظهر لك قاعدة البيانات حدد **properties** بالزرار الأيمن ثم أختر **New Table**



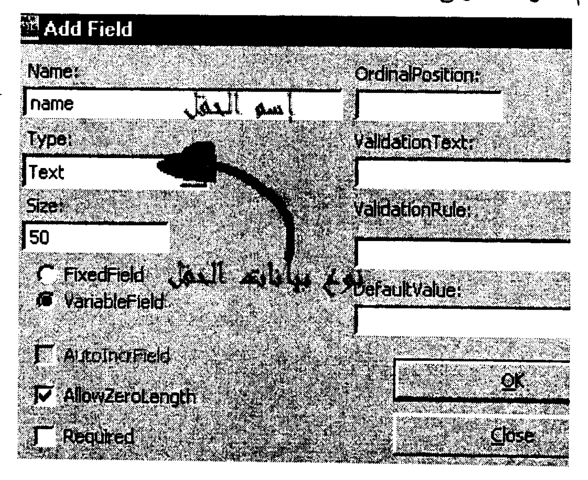
ستفتح لك نافذة تكتب فيها اسم الجدول فنذهب لإضافة الحقول كما هو مبين



ستفتح نافذة إضافة الحقول نكتب اسم الحقل في المكان المخصص ونحدد نوع بيانات الحقل هل هي رقمية أو حرفية إلخ، ونكرر العملية حتى ننهي جميع الحقول التي نريدها وفي مثالنا هذا نريد حقلين الأول اسمه **name** ونـوع بياناتـه **text** والثـاني **number** ونوع بياناته **long** .

ملاحظة: إذا كان ما سيكتب في حقل رقمي خمسة أرقام أو أقل نختار نوع البيانـات **integer**

و للأرقام الطويلة نختار **long**

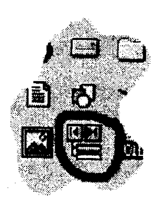


بذلك نكون قد أنشأنا قاعدة بيانات تحتوي على جدول به حقلين الاسم والرقم.

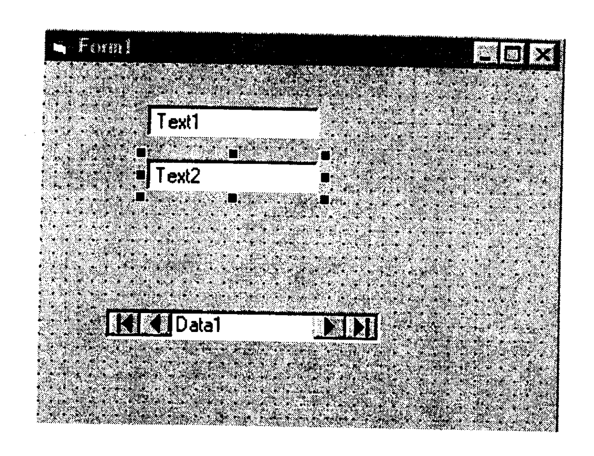
ربط قاعدة البيانات باستخدام الأداة **data** :

بعد أن أنشأنا قاعدة البيانات تأتي الخطوة التالية وهي أول خطوة في ربط قاعد البيانات سنقوم بربط القاعدة التي أنشأنها والتي تحتوي على حقلين هما اسم الطالب ورقمه:

أفتح مشروع جديد ثم أضف الأداة **data** من صندوق الأدوات كما في الشكل التالي:



نرسم الأداة على النموذج وصندوقي نص لكي نظهر فيهما الحقول



سنقوم بربط صناديق النص مع كائن البيانات وسنورد أولا خصائص كائن **data** ، من صندوق الخصائص نضبط الخصائص التالية للأداة **data**.

اسم الخاصية	قيمتها	شرح
Databasename	نكتب مسار قاعدة البيانات	و لكي نقوم بذلك نختار (..) التي بجانبها ونحدد مكان القاعدة
Recordsource	نحدد الجدول المراد ربطه	وفي حالتنا هذه فقد اسمينا الجدول **t1**

و بالنسبة لصندوقي النص نضبط خصائصها كما في الجدول التالي

اسم الخاصية	قيمتها	شرح
Datasource	**data1**	وهو اسم الكائن الذي سيربط قاعدة البيانات
Datafield	اسم الحقل المراد الربط معه	يوجد لدينا صندوقي نص لكل واحد قيمة مختلفة لكي يكون أحدهما لأسم الطالب والآخر لرقمه

هكذا تكون قد ربطت قاعدة بياناتك في برنـــامج ولتـــرى النتيجـــة قـــم بتنفيـــذ البرنامج وذلك بضغط **F5**

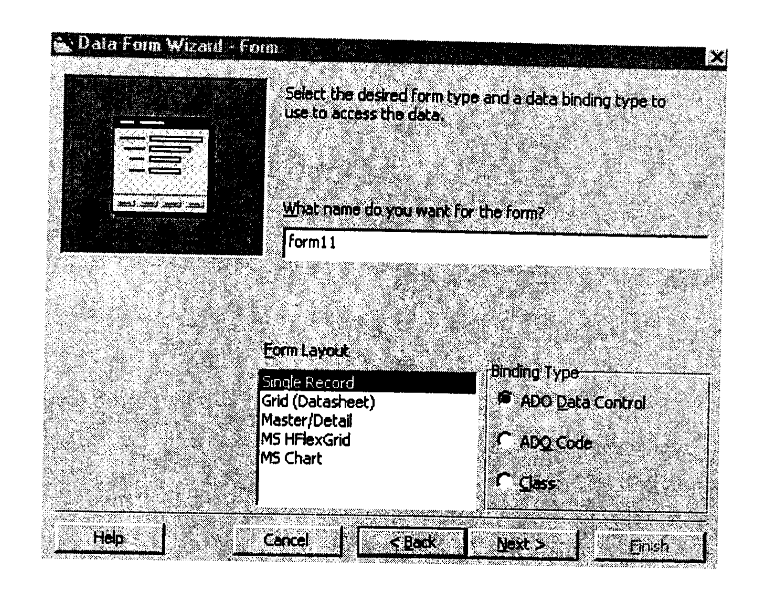
لن يظهر لك شيء لأن قاعدة البيانات فارغة ولوضع بعض الحقول، شغل منشيء قواعد البيانات وأذهب إلى

File>>Opendatabase>>microsoft access

سيفتح قاعدة بياناتك وسيظهر اسم الجدول الذي أنشأته، أنقر عليه مرتين وســـيفتح لك نافذة أضف منها ما تريد من سجلات

ربط قاعدة البيانات باستخدام المعالج السحري

أذهب إلى قائمة **Add-ins** واختر الاختيار **Add-in manager** ســـيفتح لك نافذة انقر نقرا مزدوجا على السطر **Vb 6 Data FormWizard** يجب أن تظهر العبارة التالية بجانب السطر **loaded** بعد ذلك اختر موافق واذهـــب إلى القائمة **Add-Ins** اضغط على الخيـــار التـــالي **Data Form Wizard** ستظهر لك نافذة المعالج السحري قم باختيار **Next** اختر نوع قاعدة البيانات وهو في حالتنا **Access** واختر **Next** حدد موقع قاعدة البيانات المراد ربطها وذلك بالضغط على **Browse** بعد ذلك اختر اسما للنموذج وحدد طريقة الربط والعرض كما هو مبين ثم اضغط على **Next** .



بعد ذلك حدد اسم الجدول المراد ربطه من الحقل **Record source** والحقـول المراد ظهورها من القائمة **available fields** و ذلك عن طريق تحديد الحقـل والضغط على الأسهم التي في المنتصف بعد ذلك حدد الحقل الذي تريد أن يكـون الفرز على أساسه وذلك من **column to sort by** بعد ذلك اضغط **Next** وستأتيك نافذة لتحدد الأزرار التي تريدها أن تظهر في مشروعك مثل حذف وإضافة وتحديث وما شابه حدد ما تريد ثم اختر **Next** ثم **Finish**

ستظهر لك النافذة التي أنشأناها ولكن يجب أن نجعلها الافتراضـية عنـد تشـغيل البرنـامج ولكـي تجعلـها كـذلك اذهـب إلى **..Project>>project1.propertis**ثم حدد النموذج الذي تريد أن تجعله افتراضي من القائمة **Startup Object** وهي في حالتنا **form1**

قم بتشغيل البرنامج

ثانيا: ربط قاعدة البيانات بالكود

تحتوى قواعد البيانات على جملة من الأفكار والحيل الثابتة تقريبا فمجرد تطبيقك لها من خلال الأمثلة ستتمكن من اللغة وينصح بأخذ كل شيء على محمل الجد فقد تستغرب من بعض الأمور وستقول "أن هذا ليس مستوى البرمجة بل البرمجة أعلى بكثير من ذلك"، وهذا قول خاطيء فستجد بعد انتهائك من اغلب الفصول أن البرامج الكبيرة التي كنت تعتقد أنها من المعجزات قد عملت بطرق بسيطة إن صح التعبير، لذلك اعلم أن البرمجة قائمة على أفكار وحيل ثابتة تقريبا وانت تسخرها للعمل الذي تريده وتصنع منها ما تريد.

لربط قاعدة بيانات بالكود يجب أن يكون لديك الآتي:

- قاعدة بيانات جاهزة ويفضل أن يوجد بها بعض الحقول الجاهزة.
- معرفة جيدة بإضافة أدوات التحكم الأساسية مثل الأزرار والعناوين وصناديق النص.
- تذكر أن كل قاعدة بيانات عبارة عن مجموعة من السجلات وكل سجل عبارة مجموعة عن الحقول.

عندما نريد ربط قاعدة بيانات بالكود يجب أن نعلن عن متغيرين في موديول على أنهما قاعدة بيانات وجدول ونضع القاعدة الأصلية في المتغير الذي أنشأناه لكي لا تتغير القيم الأصلية عند المعالجة المؤقتة، ونجعل الموديول الأساسي عند التحميل ثم نظهر بعد ذلك النموذج لكي يتم تعريف المتغيرين وإسناد قاعدة البيانات لهما، ولن نسند قيمه للجدول إلا في النموذج وعادة في حدث التحميل **load** سأذكر الخطوات المنطقية التي سيفعلها البرنامج بعد برمجته هي:

١. عند تنفيذ البرنامج سيذهب للموديول وسيجد متغير يحتوي على قاعدة بيانات ويتعرف عليها ويجد أمر يأمره بالانتقال للنموذج الرئيسي .
٢. سيجد عند تحميل النموذج جدول يحتوي على بيانات لكن من أين هذه البيانات؟ أنها من القاعدة التي تعرف عليها البرنامج في الخطوة الأولى.

٣. سيجد شرط يقول إذا كان الجدول يحتوي على بيانات فقم بإظهارها على صندوق النص أو القائمة حسب ما حدده المبرمج.

هذه هي الفكرة الأساسية لإظهار البيانات عند تشغيل البرنامج والآن كيف يقوم البرنامج بحفظ التغييرات الجديدة ؟

١. عندما يضغط المستخدم على زرار أضافه قم بإضافة سجل جديد

٢. عندما يقوم المستخدم بضغط زرار حفظ قم بأخذ البيانات من صناديق النص وضعها في الحقول الموازية لها واحفظها .

تطبيق علي الربط بالكود:

سيكون مشروعنا عبارة عن برنامج لحفظ اسم السلعة وسعرها، وكل ما نريده في هذا المثال هو ربط قاعدة البيانات بالكود وحفظ الإضافات الجديدة والحذف

يجب أن تجهز قاعدة بيانات وليكن اسمها **db1** وتحتوي على جدول اسمه **tb1** وهذا الجدول يحتوي على الحقول التالية :

يمكنك عمل قاعدة البيانات بواسطة **Access** أو بواسطة فيجوال بيسك وبعد أن تقوم بذلك

ملاحظات	نوع البيانات	اسم الحقل
حقل تخزين اسم السلعة	سلسلة نصية	**Name**
حقل تخزين رقم السلعة	رقمي	**Num**
حقل تخزين سعر السلعة	رقمي	**price**

- بعد أن تقوم بذلك افتح مشروع فيجوال بيسك قياسي جديد، وقبل أن تبدأ بربط قاعدة البيانات يجب أن تحدد أي طريقة ستستخدمها للربط وفي هذا المثال سنستخدم طريقة أو تقنية **DAO** ، وبعد أن حددنا الطريقة التي سوف تربط بها سنضيف المكتبة الخاصة بها وهي عبارة عن مكتبة تحتوي على تعريف للأوامر التي ستستخدمها لتقنية **DAO** ولتضيف هذه المكتبة قم بالتالي:

اذهب إلى قائمة **Project > Reference** ومن ثم حدد الاختيار **Microsoft DAO 3.51 Object Library** بعد ذلك اختر موافق

- نعود لمشروعنا، وعلينا إضافة موديول، ولذلك اذهـــب لقائمــة **Project** واضغط على **Add Module** الآن سنعرف متغيرين في الموديول واحـــد عبارة عن قاعدة بيانات والآخر عبارة عن جدول وهذه صيغة تعريف المتغيرين:

```
Public d As Database
Public t As Recordset
```

كلمة **Public** تعني انه متغير عام في المشروع ولاحظ أن **d** اسندناها كقاعدة بيانات و **t**كجدول.

- في نفس الموديول سنقوم بإسناد قاعدة البيانات الأصلية والتي اسمها **db1** إلى القاعدة الوهمية أن صح التعبير والتي عرفناها قبل قليل بـــ **d** ولعمل ذلك نكتب الإجراء التالي في الموديول

```
Private Sub main()
Set d =
DBEngine.Workspaces(0).OpenDatabase(App
.Path & "\db1.mdb")
Form1.Show
End Sub
```

لنشرح هذا الإجراء، أول شيء اسمينا هذا الإجراء باسم **main** وهذا ليس فيه خيار حيث أن هذا اسم محجوز في لغة البيسك فلا تستطيع أن تستبدله بآخر.

أول جملة في الإجراء هي جملة الإسناد ولقد بدأناها بأمر الإسناد المعـــروف **Set** ثم وضعنا القيمة التي سنسند فيها وهي **d** التي عرّفناها وقلنا سنســـند فيهـــا القاعـــدة الأصلية لكي لا تتغير قيمها إثناء المعالجة المؤقتة ثم كتبنا كلمة **DBEngine** وهي عبارة عن نوع قاعدة البيانات التي ستستخدمها وهذا هو محرك قاعدة البيانات من

نوع Access ، ثم كتبنا Workspaces(0) وهذا نوع مجال العمل، بعد ذلك وضعنا الأمر الذي سيقوم بفتح قاعدة البيانات لكي تستطيع الوصول إلى محتواهـا Opendatabase، بعد ذلك وضعنا مسار قاعدة البيانـات ولكـي تتجنـب مشكلة تغير المسار من جهاز لأخر نستخدم الدالة App.Path اي مسار المجلـد الذي يحتوي البرنامج ولاستخدام هذه الطريقة يجب أن تكون قاعدة البيانات في نفس مجلد البرنامج، ثم كتبنا اسم القاعدة وامتدادها .

في السطر الثاني من الإجراء كتبنا أمر لإظهار النموذج .

هذه الطريقة ثابتة لإسناد أي قاعدة بيانات فقط غير اسم قاعدة البيانات .

تنفيذ البرنامج

بعد قيامك بالخطوات السابقة يجب أن تنفذ البرنامج في هذه المرحلة لكي تتأكد انك تسير على الخط الصحيح وقبل ذلك تأكد من انك قمت بجعـل الموديـول في بـدأ التشغيل وذلك بالذهاب إلى ..Project > Project1.proprties بعـد ذلك حدد sub main من القائمة المنسدلة startup object بعد ذلـك شغل البرنامج بالضغط على F5 يجب أن يتم التنفيذ بطريقة صحيحة وان لم يكـن كذلك تأكد من الخطوات التالية :

١. أنك قمت بحفظ المشروع في نفس المجلد الذي فيه قاعدة البيانات
٢. أن قاعدة البيانات ليست في قيد التشغيل
٣. انك جعلت الموديول في بدأ تشغيل المشروع
٤. انك كتبت اسم قاعدة البيانات صحيحا

بعد أن تعرفنا على المتغيرات التي سنعمل عليها وربطنا قاعدة البيانات يبقى أمامنـا ربط الجدول وإظهار محتويات القاعدة للمستخدم.

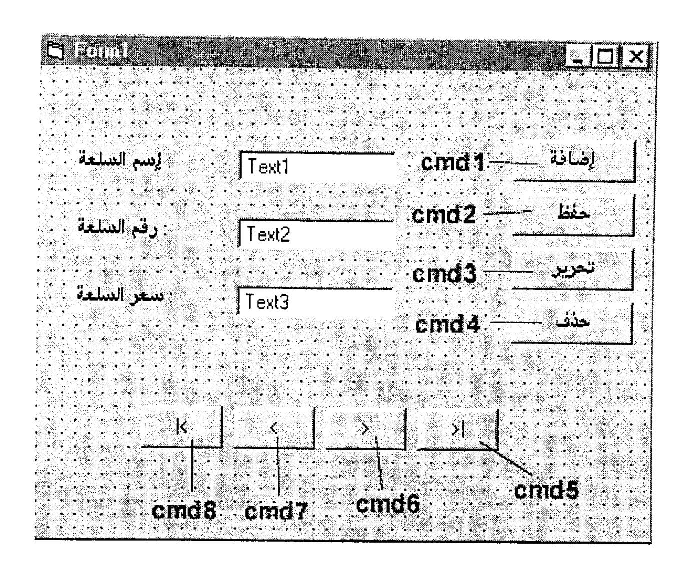
أولا سنقوم بربط الجدول الذي في قاعدة البيانات ونخزنه في المتغير الذي أنشأناه في السابق باسم t علما أن الجدول الرئيسي اسمه tb1 ولذلك اكتب الأمر التالي في حدث التحميل للنموذج load

```
Private Sub Form_Load( )
Set t = d.OpenRecordset("tb1", dbOpenTable)
End Sub
```

وضعنا ربط الجدول في حدث التحميل للنموذج وهذا ما يقوم به المحترفون لكـــي تهيء التعامل مع البيانات عند تحميل النموذج إلا إذا كان لديك سبب أخر لتخالف هذه القاعدة

بعد ذلك وضعنا جملة الربط للجدول وبدأنا بأمر الاسناد المعروف **Set** ثم اســم القيمة التي سنسند قيم الجدول فيها وهي التي عرفناها فيما قبل باسم **t** بعد ذلــك نكتب اسم القاعدة المستعار الذي اسندنا القاعدة الأصلية فيه وهو **d** ثم نكتــب الأمر الذي سيفتح لنا الجــدول لكــي نســتطيع الوصــول إلى محتــواه وهــو **Openrecordset**بعد ذلك نكتب اسم الجدول الحقيقي بين علامتي تنصيص ونكتب نوع الربط وهو **dbopentable** استخدم هذه الطريقة فقط لأنها مــن الأوأمر الثابتة ومن أنواع الربط وستستطيع التفريق بين أنواع الربط في المستقبل مع كثرة التمارين، بذلك تكون قد قمت بربط قاعدة البيانات والجدول بواسطة الكود.

لإظهار البيانات للمستخدم، يجب تصميم واجهة المستخدم بوضع صــناديق الــنص والعناوين كما في الصورة التالية وقد وضحت علــى الصــورة التســميات الــتي سنعتمدها:



لإنشاء إجراء إظهار البيانات لكي نستدعيه في كل مرة نحتاجه بدلا من إعادة كتابته سيكون على الشكل التالي:

```
Private Sub showdata( )
If t.RecordCount < 1 Then Exit Sub ' عندما يكون الجدول
فارغ اخرج من الإجراء
Text1.Text = t!Name   ' t نضع في صندوق النص الأول قيمة الجدول
حقل الاسم في
Text2.Text = t!num   'نفس الخطوة السابقة لحقل رقم السلعة'
Text3.Text = t!price   'نفس الخطوة السابقة لحقل السعر'
End Sub
```

شرح إجراء إظهار البيانات:

بدأنا بالتأكد إذا كان الجدول يحتوي على بيانات أم لا فإذا كان لا يحتوي فنخرج من الإجراء لكي لا يحدث مشاكل أثناء العرض .

بعد ذلك نظهر قيمة ما في الحقول في صناديق النص، واستخدمنا علامة التعجـــب ! لكي تفصل بين اسم الجدول المستعار واسم الحقل، يجب أن تفرق بين اسم الجدول أو القاعدة المستعار والأصلي ومتى نستخدم كل منهما .

يجب أن نضيف أمر استدعاء إجراء العرض عند تحميل البرنامج فسيكون الأمـــر في حدث التحميل هكذا:

```
Private Sub Form_Load()
Set t = d.OpenRecordset("tb1", dbOpenTable)
Call showdata
End Sub
```

كيفية التنقل بين السجلات والحفظ والتعديل والإضافة والحذف

التنقل بين السجلات

لن تحتاج للتنقل بين السجلات إلا لـــ:

السجل التالي، السجل السابق ، السجل الأول ، السجل الأخير

السجل التالي:

لكي تنتقل للسجل التالي ستحتاج لكتابة الأمر التالي في الزرار المطلوب وهو في مثالنا **cmd6**

```
Private Sub cmd6_Click( )
t.MoveNext
Call showdata
End Sub
```

لاحظ أننا استخدمنا اسم الجدول المستعار **t** ، ثم بعد ذلك اســـتدعينا الإجـــراء **showdata** الذي عملناه في الجزء السابق ليعرض البيانات في السجل التـــالي وهذه الطريقة تنطبق على جميع أنواع التنقل القادمة .

السجل السابق:

هذا الكود نفس طريقة الكود السابق مع تغـيره إلى MovePrevious لا تنسى ضع الكود في المكان المناسب وهو هنا cmd7 .

```
Private Sub cmd7_Click( )
t.MovePrevious
Call showdata
End Sub
```

السجل الأول:

للانتقال للسجل الأول اكتب الكود التالي في زرار الأمر cmd8

```
Private Sub cmd8_Click( )
t.MoveFirst
Call showdata
End Sub
```

السجل الأخير

للانتقال للسجل الأخير اكتب الكود التالي في زرار الأمر cmd5

```
Private Sub cmd5_Click()
t.MoveLast
Call showdata
End Sub
```

هذه كل أوامر التنقل التي تحتاجها، لكن ستواجهك مشكلة، فمثلا عندما تريـد أن تنتقل للسجل التالي وأنت في السجل الأخير لا يوجد سجل تالي لذلك سـيتوقف البرنامج، ونفس الشيء عند الانتقال للسجل السابق وأنت في السجل الأول فـلا يوجد سجل سابق فسيتوقف البرنامج، لذلك سنقوم بإضافة جملة شرطية للتأكـد إذا كان السجل الأخير أو الأول حسب الحالة ثم نقوم بوضع أمر Movefirst أو movelast حيث أن هذين الأمرين لا يتأثران سواء كان هنـاك سـجل أو لا وطريقة استخدامهم هكذا

ملاحظة:

نستخدم الدالة **EOF** لمعرفة آخر سجل في الجدول **End Of File**، ونستخدم الدالة **BOF** لمعرفة أول سجل في الجدول**Begin Of File**

سنضع التالية في زرار الانتقال للتالي فإذا كان هذا آخر سجل إذا انتقل للسجل التالي الجملة الشرطية

```
If t.EOF Then t.MoveLast
```

الانتقال للسابق نضع شرط إذا كان هذا أول سجل إذا انتقل للسجل الأول وكذلك في زرار

```
If t.BOF Then t.MoveFirst
```

لذلك سنضيف الجملتين السابقتين لكود الانتقال للتالي والانتقال للسابق فيصبح كود الانتقال للتالي هكذا:

```
Private Sub cmd6_Click( )
t.MoveNext
If t.EOF Then t.MoveLast
Call showdata
End Sub
```

وكود الانتقال للسابق:

```
Private Sub cmd7_Click()
t.MovePrevious
If t.BOF Then t.MoveFirst
Call showdata
End Sub
```

عمليات السجلات:

سندرس عمليات السجلات من حذف وإضافة وتعديل

إضافة سجل ، حفظ سجل ، تعديل سجل، حذف سجل

إضافة سجل:

الأمر التالي يقوم بإضافة سجل ونضيف عليه أوامر لتمسح ما في صناديق النص لتهيئتها للإضافة:

```
Private Sub cmd1_Click( )
t.AddNew 'جديد إضافة سجل
' الخطوات التالية لمسح ما في صناديق النص لتهيئتها للإضافة وهي خطوة لتعطي طابع الاحتراف
Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
End Sub
```

حفظ سجل

لحفظ سجل يجب وضع القيم التي في صناديق النص في الحقول التي توازيها في الجدول، ولاحظ أنه سوف يعطيك رسالة خطأ عندما تقوم بالحفظ دون أن تقوم باختيار تعديل سجل أو إضافة سجل لذلك يفترض بك أن تجعل زرار الحفظ في حالة التمكين فقط عندما يضيف المستخدم سجلا أو يختار تعديل سجل، كذلك يجب عليك استخدام الدالة **Val** عند حفظ الحقول الرقمية لتجنب المشاكل عندما يتركه المستخدم فارغا، ويجب عليك تحديث الجدول بعد عملية الحفظ لتكمل العملية وهذا هو الكود المطلوب:

```
Private Sub cmd2_Click( )
' نقوم في الخطوات التالية بنقل ما في صناديق النص إلى الحقول التي توازيها في قاعدة البيانات
t!Name = Text1.Text
t!num = Val(Text2.Text) ' لاحظ إننا استخدمنا هذه الدالة لكي يتم
Val قبول الحقل في حالة كونه فارغ لان هذا حقل رقمي
t!price = Val(Text3.Text)
t.Update
End Sub
```

تعديل سجل:

اخبر البرنامج انك تريد التعديل بهذا الكود

```
Private Sub cmd3_Click()
t.Edit ' تسمح هذه الخاصية بتعديل البيانات في الحقل
End Sub
```

حذف سجل:

عملية حذف السجل سهلة، ولكن ماذا بعد أن تحذف السجل ؟ بالطبع يجب أن تعرض السجل التالي، وان تراعي المشاكل التي تواجهك عند انتقالك للسجل التالي فقد لا يكون هناك سجل تالي وقد شرحنا كيف تتفادى هذه المشكلة في هذا ، وهذا هو الكود اللازم:

```
Private Sub cmd4_Click()
t.Delete 'لحذف سجل
t.MoveNext 'للانتقال للسجل التالي بعد الحذف
If t.EOF Then t.MoveLast 'يحل هذا مشكلة عدم وجود سجل تالي
الإجراء
End Sub
```

لغة الاستفسارات SQL

لن تستطيع تجاهل عند تعلم قواعد البيانات هو لغة الاستفسار أو الاستعلام والتي يعبر عنها بـــSQL ، لكن ما هي فائدة هذه اللغة؟ لنفرض أن لديك موظفين بالمئـــات وتريد أن تحدد جنسية معينه منهم وتحصرهم لن تستطيع فعل ذلك إلا بلغة الاستعلام حيث يقوم بجمع الحقول التي تساوي الشرط الذي شرطيه وهو جنسيتهم، ســـنأخذ شرحا مفصلا في هذا عن الأساسيات وتطبيقات عليها.

أولا ما فائدة هذه اللغة ؟ كما ذكرنا في المقدمة أن فائدتها في حصر القيم التي تطابق القيم المعطاة من قبل المستخدم وتتميز بالسهولة والدقة والسرعة كما يمكنك التحكم بها بصوره كبيرة ويمكنك إدخال أكثر من شرط في الاستعلام الواحد، لنفـــرض أن لدينا جدول اسمه Tb وفيه حقلين باسم name و number وتحتـــوي هـــذه الحقول على قيم مدخلة كما في الجدول التالي :

Name	سامي	سامي	صالح
Number	445	534	444

ونريد أن نستخرج الأسماء التي أرقامها ٤٤٤ فسنحتاج إلى جملة استعلام بسيطة على الشكل التالي :

SQL = "select name from Tb where number = 444"

من الجدول **name** أن يقوم بتحديد قيم الحقل في الجملة السابقة أمرنا البرنامج **Tb**عندما تكون قيم الحقل **number** تساوي **444** ، ووضعنا ذلك في متغير اسمه **SQL**

طرق الاستعلام

للاستعلام عن جميع الحقول في جدول بدون شرط استخدم النجمة لتعبر عن جميع الحقول ولا تضع شرطا فى مثل هذا الاستعلام :

SQL = "select * from Tb "

إذا أردت أن تستفسر بدون شرط اكتب اسم الحقلين بينهما فاصلة ولا تضع شرطا عن قيم حقلين كالتالي :

SQL = "select name, number from Tb "

استخدم الصيغة التالية إذا أردت أن تستفسر عن حقل معين بشرط ما

SQL = "select name from Tb where number = 444 "

يمكنك استخدام صيغ المقارنة < أو > أو => أو =< أو <> بدلا من =

عندما تريد الاستعلام بأكثر من شرط ضع بين الشرطين عبارة **And** أو **OR** واستخدامها حسب حاجتك كالتالي :

SQL = "select name from Tb where number = 444 or number = 555"

استخدام لغة الاستفسار في تطبيق فيجوال بيسك، من حيث كيفية الاستفسار عـــن قيمة مدخلة من قبل المستخدم وكيفية وضع نتائج البحث في **listbox** و **combo box**

وكيف نتعامل مع جملة **SQL** في البرنامج ؟

سنقوم بالاستعلام وذلك بكتابة جملة الاستعلام في المكان الذي تريده وغالبا ما يكون في زرار أو عند حدث التحميل، بعد ذلك سنحصل على نتائج الاستعلام ولكي نقوم بقراءها وإظهارها يجب أولا أن نخزنها في جدول، وهنا فائدة الجدول المستعار كمـــا حيث سنقوم بتخزين الناتج فيه ، بعد ذلك سنقوم بعرض النتـــائج مـــن الجـــدول، سنفرض أن لدينا الجدول التالي:

الجدول اسم**tb**

Name	Num
ahmad	1442
Saleh	5425
Sami	1442

لكي نستعلم عن الأسماء التي أرقامها ١٤٤٢ نكتب الجملة التالية:

SQL = "select name from Tb where num = 1442"

ذلك تأتي الخطوة التالية وهي تخزين النتـــائج في هكذا نكون قد قمنا بالاستعلام بعد الجدول المستعار والذي نقوم بتعريفه عادة في الموديول كما قمنا بـــه في الـــدروس السابقة ولنفرض أن اسمه **T** وان اسم المتغير الذي قمنا بتخزين قاعدة البيانات فيه **D** والتي كنا نطلق عليها القاعدة المستعارة سنكتب الأمر التالي لكي نخـــزن نتـــائج الاستعلام في الجدول:

Set T=D.openrecordset(SQL,dbopendynaset)

لا حظ أننا استخدمنا الأسماء المستعارة للجدول وقاعدة البيانات وهي المتغيرات التي أعلنا عنها في البداية ، لقد قمنا في الأمر السابق بتخزين قيم الاستعلام في الجدول حيث كتبنا SQL وهي قيمة متغير الاستعلام الذي توجد فيه القيم، بعد ذلك قمنا بوضع نوع الربط المراد وبعد ذلك لأظهار البيانات يمكنك إظهارها في عدة أشكال فيمكنك وضعها في صناديق نص أو قوائم سواء List أو COMBO ولكل واحد طريقة تختلف نوعا ما عن الآخر

صندوق النص **:Text Box**

لأظهار النتائج في صندوق النص لن تحتاج إلا لإجراء الإظهار كما في السابق وتقوم بكتابته بعد أمر تخزين نتائج الاستعلام في الجدول.

List Box:

لتضع النتائج في قائمة **ListBox** تحتاج لهذا الكود:

```
For i = 1 To n
List1.AddItem TB!name
TB.MoveNext
Next i
```

قمنا بعمل تكرار بعدد الحقول لكي يتم تعبئة جميع البيانات في الجدول، بعد ذلك كتبنا اسم القائمة وهو List1 ثم استخدمنا خاصية إضافة عنصر وبعد ذلك نكتب اسم الجدول الذي سوف نأخذ البيانات منه ثم اسم الحقل المراد يفصل بينهما علامة تعجب، بعد ذلك خطوة مهمة وهي الانتقال للسجل التالي لكي يقوم بتعبئة القيم الأخرى وإذا لم تضع هذا الأمر سوف تكون جميع الأسماء اسم الحقل الأول.

ComboBox

لا تختلف طريقة تعبئة هذه القائمة عن الطريقة السابقة أبدا قم بنفس الخطوات

استقبال المدخلات من المستخدم:

لن تستفيد حقيقيا من البرامج التي تعملها إذا لم تعرف طريقة استقبال القيم من المستخدم ثم عمل الاستعلام عليها لأنه من غير المنطقي أن يقوم المستخدم بالرجوع إليك عند كل عملية استعلام، وطريقة الاستقبال هنا لها طريقة خاصة نوعا ما فغالبا ما تقوم باستقبال القيم من المستخدم من صندوق نص **Text Box** والمشكلة التي تواجهنا بين المدخلات النصية والرقمية ويعتمد ذلك على نوع البيانات للحقل المعني ادرس الأمثلة التالية:

افرض أنك قمت بعمل قاعدة بيانات فيها حقلين الاسم **name** وحددت نوع البيانات لهذا الحقل بأنها نصية والحقل الآخر الرقم **num** وحددت البيانات له بأنها رقمية، وقمت بربط القاعد مع البرنامج ونريد أن نستعلم عن الرقم للشخص الذي يحدده المستخدم سنقوم في هذه الحالة بإنشاء صندوق نص لكي يدخل المستخدم الاسم الذي يريده ثم ننشأ زرار يقوم المستخدم بالضغط عليه لتتم عملية الاستعلام، وستكون جملة الاستعلام عادية وبدلا من أن نضع القيمة التي سنستعلم عنها سنضع القيمة التي ادخلها المستخدم في صندوق النص من المتوقع أن يكون الكود المطلوب للاستعلام في هذه الحالة التالي:

SQL = "select num from tb where name = text1.text "

هذه هي الجملة المتوقعة لكنها خطأ، بهذه الطريقة يكون الاستعلام عن عدد لانهائي من الخانات وهذه أن صح التعبير خطأ في صندوق النص ولكي نتلافى هذا الخطأ نقوم بما يسمى بالحصر على أساس نوع البيانات في **name** في حالتنا هذه نوع البيانات حرفية لذا تكون طريقة الحصر على الشكل التالي :

' " & text1.text & " '

وضعنا علامتي تنصيص مزدوجة عليها يفصل بينها علامة الجمع **&** ثم قمنا بحصرهم جميعا بعلامة تنصيص مفردة، الخطوة الأخيرة وهي وضع علامة التنصيص المفردة تكون للحقول الحرفية ولا نحتاجها في الحقول الرقمية.

افرض أن المطلوب الآن هو العكس، بحيث يقوم المستخدم بإدخال الرقم ويقوم البرنامج بالاستعلام عن الأسماء التي تحمل هذا الرقم سيكون الكود نفس السابق مع تغيير اسم الحقل بطبيعة الحال وأيضا عدم وضع علامة التنصيص المفردة كالتالي:

SQL = "select num from tb where name = '" & text1.text & "' "

سيوضح ما يلي بعض النقاط المهمة في هذا بالنسبة لاستقبال المدخلات من المستخدم.

نستخدم الطريقة التالية عندما تكون بيانات الشرط رقمية أو عددية ويحدد ذلك من ينشأ قاعدة البيانات

" & text1.text & "

نستخدم الطريقة التالية عندما تكون بيانات حقل الشرط من نوع البيانات الحرفية

'" & text1.text & "'

لغة الاستفسارات SQL

استخدام الاستعلام في التطبيق وهذه أهم نقطة في لغة الاستعلام، والآن سندرس تطبيق متقدم في لغة الاستعلام وهو الاستعلام من أكثر من جدول، أي نقصد مثلا أن يطلب منا الاستعلام عن الأشخاص الذين يعملون ضمن قسم المحاسبة، الشرط هنا هو العمل في قسم المحاسبة لكن الشرط غير موجود في الجدول ؟ إذا كيف سنقوم بالاستعلام ؟ سنقوم بالاستعلام عن الشرط من جدول آخر ثم نرجع النتيجة ويستعلم في الاستعلام الأساسي عليها وبذلك يكون لدينا استعلامين، ولكن يجب أن يكون هناك حقل مشترك بين الجدولين ، سيتضح ذلك في السطور التالية

قبل أن نبدأ افرض أن لدينا الجدولين التاليين

tb1

deptno	job	sal	Name
3	محاسب	6000	جميل
1	مدير	12000	سامي

2	مبرمج	5500	بدر
1	مهندس	8500	مهند
3	سكرتير	5000	صالح

tb2

deptn	deptname
1	الهندسة
2	الكمبيوتر
3	المحاسبة

نريد الاستعلام عن أسماء الموظفين الذين ينتمون لقسم ما ولنفرض أننا نريد من ينتمي لقسم الهندسة، ستكون صيغة الاستعلام على الشكل التالي " حدد حقل الاسم من الجدول **tb1** عندما يكون اسم القسم الهندسة "، لكن هناك مشكلة حيث لا يوجد حقل باسم القسم في نفس الجدول لكن يوجد اسم القسم في جدول آخر، ويوجد في الجدول الذي لدينا رقم القسم لكننا نريد البحث على اسم القسم وليس رقمه، إذا يجب أن نحصل على اسم القسم من الجدول الآخر، وللحصول على معلومات من جدول آخر يجب تحقق شرط مهم وهو وجود حقل مشترك بين الجدولين وهذا ينطبق في حالتنا، إذا ستكون صيغة الاستعلام على الشكل التالي: " حدد حقل الاسم من الجدول **tb1** عندما يكون اسم القسم يساوي اسم القسم من الجدول **tb2** عندما يكون رقم القسم ١ "

لقد وضعنا الاستعلام بالصورة المطلوبة حيث كنا نفتقد للشرط في الاستعلام الأول لعدم وجوده في نفس الجدول لذا قمنا بعمل استعلام آخر فرعي لنحصل منه على الشرط، سنحتاج الى الكود التالى بلغة الاستعلام:

```
SQL = "select name from Tb1 where deptname =
(select deptname form tb2 where deptno = 1) "
```

لماذا نقوم بهذه العملية الطويلة مع أنه بإمكاننا أن نشترط شرط على أســاس رقــم القسم الذي يتوفر لدينا حيث نعرف أن قسم الهندسة رقمه ١ ؟ الجواب أنه ليس في كل مرة ستعرف ذلك حيث توجد استعلامات معقدة لا يمكنك أن تقوم بحســاب الشرط بنفسك لكي تضع استعلام واحد، حيث يجب في بعض التطبيقات استخدام الاستعلام المركب وقد يصل لأكثر من استعلام لحل وإيجاد قيمة استعلام واحد في الاستعلام المتفرع من استعلام لا نضع علامتي تنصيص مزدوجة على الشرط النصي بل نضع علامة تنصيص مفردة مثل هذه'

تطبيقات على لغة الاستفسارات SQL

تطبيقات على الاستعلام عن الحروف الأولى من حقل ما، فمثلا في الجامعات يكتبــون الحروف الثلاثة الأولى من اسم المقرر، وغيرهم أيضا يحتاج لمثل هذه الخاصية، وسوف ندرس كيف نقوم باستخراج الأحرف الثلاثة الأولى من اليسار لجميع القيم في حقل معين، الكود الذي سيقوم بهذه العملية هو التالي:

```
SQL = "select left$(name,3) as k from tb1"
```

استخدمنا الدالة **left$** لتحديد الحروف من حقل معين والذي اسمه في المثال السابق **name** وقد قمنا بتخزين القيم في المتغير **k** ويمكنك تخزينها في أي متغير تريد، الجدول في حالتنا اسمه**tb1** ، نقوم بعد ذلك بإظهار النتائج

ولتحديد القيم التي تحتوي على حرف معين سواء في منتصف الكلمة أو في أولهــا أو أخرها، هذا التطبيق مهم وله استخدامات عديدة في البحث خصوصا عندما تعرف أن اسم الطالب يحتوي على حرف معين لكن لا تعرف اسم الطالب وأين موقع الحرف في اسمه، طريقة الاستعلام التالية سوف تبين لنا كيفية استخدام هذه الطريقة، الكود هو:

```
SQL = "select name from Tb1 where name like '*"
& text1.text & "*'"
```

لا حظ أننا قمنا باستخدام الجملة **Like** وهي بمعنى التشابه وليس شرطا التساوي بعد ذلك وضعنا الشرط وهو ما يدخله المستخدم في صندوق النص ونظرا لأننا وضعنا علامتين نجمتين في الشرط وهي تستخدم في إعلام البرنامج بأنه سيحدد أي قيمة تشبه القيمة المدخلة في أي موقع منها

إن الاستعلام يعيد القيم مرتبة على أساس ترتيب إدخالها، لكن أحيانا تحتاج لأن تقوم بترتيب النتائج ترتيبا تصاعديا أو تنازليا ولذلك نستخدم الأمر **Order By** ويكون شكل الاستعلام على الطريقة التالية:

SQL = "select name from Tb1 where name like '*" & text1.text & "*' order By name Asc"

بعد كتابة الأمر **Order by** اكتب اسم الحقل الذي تريد أن يكون الفرز والترتيب عليه وهو في حالتنا **name** بعد ذلك اكتب طريقة الفرز التي تريدها تصاعدي أي من الألف للياء ونستخدم لذلك أمر **Asc** ، مع العلم أن عند عدم وضع **Asc** فإن البرنامج يقوم تلقائيا بفرزه تصاعدي، إما الفرز التنازلي من الياء إلى الألف فنستخدم الأمر **Desc** بدلا من **Asc**

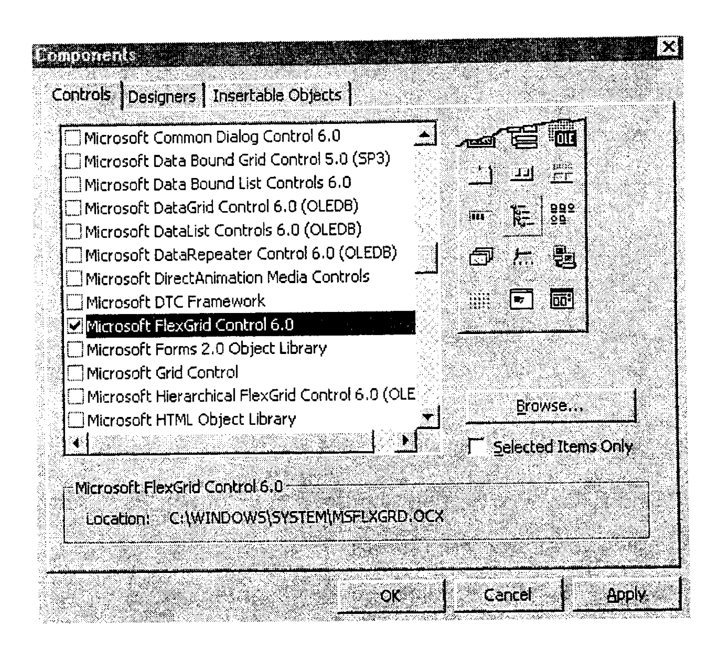
التعامل مع **Grid Flex**

أداة **FlexGrid** إحدى أدوات إظهار البيانات في شكل شبكي أو جدولي سندرس كيفية وضع نتائج استفسار فيها وإظهاره للمستخدم وكذلك بعض الأفكار في التعامل مع هذه الأداة .

فيما يلي الصورة التي نريد أن نصل إليها في النهاية حتى تتعرف على ما سنعمل عليه انظر الصورة التالية:

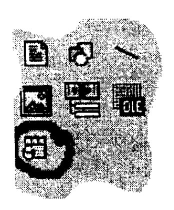
name	job	dgree	sal	age
mohammad	manager	dr	20000	45
sami	engineer	dr	15000	30
fahad	computer	dr	18000	33
khalid	clean	nothing	2000	27
ahmad	engineer	master	10000	38
faisal	computer	master	11000	32

لكي نبدأ أضف مشروع جديد وانشأ أو استخدم أي قاعدة بيانـات واربطهـا في البرنامج ، وسنفترض أن قاعدة البيانات التي سنتعامل معها اسمها **db** وفيها جدول باسم **tb** يحتوي على الحقول التالية **name** و**job** و **age** سـنكتفي بثلاثـة حقول، بعد ذلك يجب أن تضيف أداة هذه الشبكة ولـذلك، اذهـب إلى قائمـة **project >> combonent**أو اضغط على **Ctrl+T** ستفتح لك قائمـة المكونات اختر منها المكون التالي وضـع بجانبـه علامـة صـح **Microsoft FlexGrid Control 6.0**كما هو موضح في الصورة



بعد تحديد المكون الذي في الصورة السابقة اضغط **Ok** لتضيف المكون للمشروع وتعود للمشروع .

بعد ذلك اختر الأداة الخاصة برسم **FlexGrid** كما هو مبين في الصورة التاليـــة وارسم على النموذج مساحة معقولة تأخذ تقريبا ثلث النموذج.



اضغط بالزرار الأيمن على الشبكة التي قمت برسمها واختر **Properties** ستفتح لك نافذة الخصائص وفي علامة التبويب **General** يمكنك اختيار عدد الصـــفوف من الخاصية **Rows** وعدد الأعمدة من الخاصية **Cols** ، كما يمكنك تحديد عدد

الاعمدة التي تكون بلون داكن كمفتاح للشبكة من الخاصية **Fixed Cols** أو الصفوف التي بلون داكن من الخاصية **Fixed Rows** .

وللتحكم بخصائص **FlexGrid** بواسطة الكود، يجب أن تعلم أن جميع الأوامر التي تكتب لهذه الأداة غالبا ما تكتب في حدث التحميل للنموذج **Load** ، ومن الخصائص الهامة نجد التحكم في عرض الصفوف مع العلم بأن الصفوف مرقمة من اليسار إلى اليمين ابتداء من الصفر وإذا أردنا مثلا أن نغير عرض العمودين رقم ٠ و٣ وأن يكون عرضهما **2000** و **1000**على التوالي فإننا سنحتاج إلى كتابة الكود التالي :

```
MSFlexGrid1.Colwidth(0) = 2000
MSFlexGrid1.Colwidth(3) = 1000
```

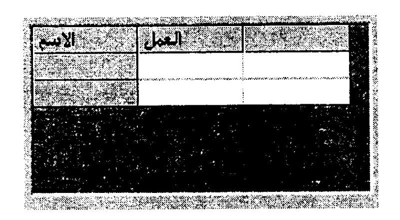
مع العلم بأن اسم الأداة في الكود السابق هو**MSFlexGrid1** ، لاحظ أننا نقوم بحصر رقم الصف بين قوسين

و لوضع عناوين الحقول على الأعمدة في هذه الشبكة لكي نفرق بين الحقول، نحدد الخلية بالصف والعمود بعد ذلك نكتب النص المراد، مثلا نريد أن نكتب عنوان الحقل الأول وهو "الاسم " في العمود الأول وعنوان الحقل الثاني وهو " العمل " في العمود الثاني، انظر للكود التالي وستعرف :

```
MSFlexGrid1.Row = 0
MSFlexGrid1.Col = 0
MSFlexGrid1.Text = "الاسم "
MSFlexGrid1.Col = 1
MSFlexGrid1.Text = "الوظيفة"
```

لاحظ أننا في أول سطرين من الكود السابق قد قمنا بتحديد أول خلية على اليسار في الزاوية العليا وذلك بتحديد إحداثياتها وهي الصف الأول والعمود الأول، ثم قمنا بكتابة العنوان بالخاصية **Text** ، مع العلم بأن البرنامج يقوم بتخزين إحداثيات آخر خلية تقوم بكتابتها، لذلك وعندما نكتب إحداثيات النقطة الثانية ستستفيد من

هذه الميزة فسنكتب إحداثي العمود وهو ١ لأن الاحداثي السابق للصف هو ٠ وهو مخزن فماذا نكتبه مرة أخرى مادمنا لا نحتاج لتغييره، بعد ذلــك نكتــب العنــوان باستخدام الخاصية **Text** كما فعلنا مع الأولى ونستمر بهذه العملية إذا احتجنا إلى عناوين أخرى، وستظهر العناوين التي عملناها بالكود السابق كما الشكل التالي:



و لتعبئة البيانات من جدول أو استعلام:

نشيء جدول وهمي لكي نخزن فيه البيانات مؤقتا كما هو حال الرام **RAM** في الجهاز لتخزين البيانات لنتعامل معها وبعد إغلاق الجهاز تــذهب المعلومــات، أن ديناميكية تعبئة البيانات بسيطة ونحتاج أولا وضع عناوين الأعمدة كما في الفقــرة السابقة ويكتب ذلك غالبا في حدث التحميل **Load** بعد ذلك يأتي دور تعبئــة السجلات، وأضف سطر يتأكد من وجود بيانات في الجدول لكي لا تواجه مشاكل، بعد ذلك احسب عدد السجلات في الجدول ولذلك يجب أولا أن نمر على الجــدول إلى النهاية ثم نعود إلى البداية لكي نستطيع إيجاد عدد السجلات بواسطة الخاصــية **Count**بعد ذلك نعد تكرار على عدد السجلات ونضع داخله كود يقوم بتعبئة البيانات بنفس الطريقة التي قمنا فيها بوضع عناوين الأعمدة ولكن يكون المصدر هو الجدول، لاحظ أن اسم الجدول الوهمي **TB** ويوجــد بــه الحقــول **Name** و **Job**، مع ملاحظة أن اسم **FlexGrid** في الكود التالي هو**flx1** .

If tb.RecordCount < 1 Then Exit Sub ' لكي نتأكد من وجود سجلات في الجدول أو نخرج في عدمها

tb.MoveLast ' نتحرك إلى الحقل الأخير

```
tb.MoveFirst  ' نتحرك إلى الحقل الأول
n = tb.RecordCount  ' نخزن عدد السجلات في المتغير n
For i = 1 To n
    flx1.Row = i  ' رقم الصف حسب رقم التكرار
    flx1.Col = 0  ' رقم العمود ثابت لأننا سوف نضع جميع القيم في للحقل
في عمود واحد
    flx1.Text = tb!Name  ' نضع قيمة الحقل في الخلية المحددة
    flx1.Col = 1
    flx1.Text = tb!job
    tb1.MoveNext  ' نتحرك للسجل التالي
Next i

End Sub
```

من المهارات الممتازة معرفة كيفية جعل عدد الصفوف يتغير حسب عدد السجلات ولذلك اجعل خاصية عدد الصفوف تساوي عدد السجلات في الجدول زائدا واحد، وقمنا بهذه الزيادة لأنها تعبر عن صف عناوين الحقول وضع هذه الجملة في المكان المناسب بعد حساب عدد الحقول في الجدول ويوضح الكود التالي موقع الجملة من الكود السابق:

```
If tb.RecordCount < 1 Then Exit Sub
 tb.MoveLast
tb.MoveFirst
n = tb.RecordCount
flx1.Rows = n + 1
For i = 1 To n
   flx1.Row = i
   flx1.Col = 0
   flx1.Text = tb!Name
   flx1.Col = 1
   flx1.Text = tb!job
```

```
        tb.MoveNext
    Next i

End Sub
```

المحتويات